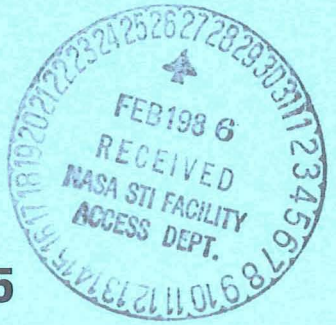
SOFTWARE ENGINEERING LABORATORY SERIES SEL-78-202

FORTRAN STATIC SOURCE CODE ANALYZER PROGRAM (SAP) USER'S GUIDE (REVISION 2)



APRIL 1985

(NASA-CR-C8586) FORTRAN STATIC SOURCE CODE ANALYZER PROGRAM (SAP) USER'S GUIDE: REVISION 2 (NASA) 131 p CSCL 09B
N86-19964
Unclas
00/61 05489

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

SOFTWARE ENGINEERING LABORATORY SERIES SEL-78-202

FORTRAN STATIC SOURCE CODE ANALYZER PROGRAM (SAP) USER'S GUIDE (REVISION 2)

APRIL 1985

NASA
National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

FOREWORD

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) and created for the purpose of investigating the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1977 and has three primary organization members:

NASA/GSFC (Systems Development and Analysis Branch)
The University of Maryland (Computer Sciences Department)
Computer Sciences Corporation (Flight Systems Operation)

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effect of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document.

Contributors to this document include

William Decker	(Computer Sciences Corporation)
Wayne Taylor	(Computer Sciences Corporation)

Other contributors include

Suellen Eslinger	(Computer Sciences Corporation)
Frank McGarry	(Goddard Space Flight Center)
Phil Merwarth	(Goddard Space Flight Center)
Thomas Manteufel	(Goddard Space Flight Center)

Single copies of this document can be obtained by writing to

Frank E. McGarry
Code 552
NASA/GSFC
Greenbelt, Maryland 20771

ABSTRACT

This document presents the FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 2). SAP is a software tool designed to assist Software Engineering Laboratory (SEL) personnel in conducting studies of FORTRAN programs. SAP scans FORTRAN source code and produces reports that present statistics and measures of statements and structures that make up a module. This document provides instructions for operating SAP and contains information useful in interpreting SAP output. It is a revision of the previous SAP user's guide, SEL-78-102, and is the result of program modifications to update several reports, additional software quality analysis, and recognition of all statements described in the FORTRAN dialects of the host VAX-11/780 and IBM 4341 computers.

8721 PAGE iv INTENTIONALLY BLANK

TABLE OF CONTENTS

Section 1 - Introduction 1-1

Section 2 - SAP Program Operation. 2-1

2.1 SAP Input Files 2-1

2.1.1 Reading Foreign Tapes on the VAX-11/780 2-1
2.1.2 Reading Foreign Tapes on the IBM 4341. . 2-2

2.2 SAP Interactive Execution on the VAX-11/780 . . . 2-2
2.3 SAP Batch Execution on the IBM 4341 2-7
2.4 SAP Output. 2-11

2.4.1 Global Statistics File 2-11
2.4.2 Module Statistics File 2-14
2.4.3 Listing File 2-14
2.4.4 Sequential Output File 2-18

Section 3 - Overview of SAP. 3-1

3.1 SAP Processing Overview 3-1
3.2 Statistics Gathered 3-2
3.3 Keywords File 3-25
3.4 Statistical Weights File. 3-25
3.5 Halstead's Software Science Metrics 3-29
3.6 McCabe's Complexity Measure 3-30

Appendix A - FORTRAN Statement Analysis. A-1

A.1 Discussion. A-1
A.2 FORTRAN Statements. A-6

A.2.1 ACCEPT Statement A-6
A.2.2 Arithmetic Statement Function Definition A-6
A.2.3 ASSIGN Statement A-6
A.2.4 ASSIGNMENT Statement A-6
A.2.5 AT Statement A-6
A.2.6 BACKSPACE Statement. A-6
A.2.7 BLOCK DATA Statement A-7
A.2.8 BYTE Statement A-7
A.2.9 CALL Statement A-7
A.2.10 CHARACTER Statement. A-7
A.2.11 CLOSE Statement. A-7
A.2.12 COMMON Statement A-8
A.2.13 COMPLEX Statement. A-8
A.2.14 CONTINUE Statement A-8
A.2.15 DATA Statement A-8

PAGE vii INTENTIONALLY BLANK

Appendix A - Cont'd)

A.2.16	DEBUG Statement.	A-8
A.2.17	DECODE Statement	A-9
A.2.18	DEFINEFILE Statement	A-9
A.2.19	DELETE Statement	A-9
A.2.20	DIMENSION Statement.	A-9
A.2.21	DISPLAY Statement.	A-9
A.2.22	DO Statement	A-10
A.2.23	DOUBLECOMPLEX Statement.	A-10
A.2.24	DOUBLEPRECISION Statement.	A-10
A.2.25	DOWHILE Statement.	A-10
A.2.26	EJECT Statement.	A-11
A.2.27	ELSE Statement	A-11
A.2.28	ELSEIF Statement	A-11
A.2.29	ENCODE Statement	A-11
A.2.30	END Statement.	A-11
A.2.31	END DEBUG Statement.	A-11
A.2.32	ENDDO Statement.	A-12
A.2.33	ENDFILE Statement.	A-12
A.2.34	ENDIF Statement.	A-12
A.2.35	ENTRY Statement.	A-12
A.2.36	EQUIVALENCE Statement.	A-13
A.2.37	EXTERNAL Statement	A-13
A.2.38	FIND Statement	A-13
A.2.39	FORMAT Statement	A-13
A.2.40	FUNCTION Statement	A-13
A.2.41	GOTO Statement	A-14
A.2.42	IF Statement	A-14
A.2.43	.IF Statement.	A-14
A.2.44	IMPLICIT Statement	A-14
A.2.45	INCLUDE Statement.	A-15
A.2.46	INQUIRE Statement.	A-15
A.2.47	INTEGER Statement.	A-16
A.2.48	INTRINSIC Statement.	A-16
A.2.49	LOGICAL Statement.	A-16
A.2.50	NAMELIST Statement	A-17
A.2.51	OPEN Statement	A-17
A.2.52	PARAMETER Statement.	A-18
A.2.53	PAUSE Statement.	A-18
A.2.54	PRINT Statement.	A-18
A.2.55	PROGRAM Statement.	A-18
A.2.56	READ Statement	A-19
A.2.57	REAL Statement	A-19
A.2.58	RETURN Statement	A-19
A.2.59	REWIND Statement	A-20
A.2.60	REWRITE Statement.	A-20
A.2.61	SAVE Statement	A-20

TABLE OF CONTENTS (Cont'd)

Appendix A - Cont'd)

A.2.62 STOP Statement A-20
A.2.63 SUBROUTINE Statement A-21
A.2.64 TRACE OFF Statement. A-21
A.2.65 TRACE ON Statement A-21
A.2.66 THEN Statement A-21
A.2.67 TYPE Statement A-21
A.2.68 UNLOCK Statement A-22
A.2.69 VIRTUAL Statement. A-22
A.2.70 WAIT Statement A-22
A.2.71 WRITE Statement. A-22

Appendix B - Halstead's Measures in SAP. B-1

B.1 Introduction. B-1
B.2 Counting Halstead Operators B-2

B.2.1 Delimiter Operators. B-2
B.2.2 Keyword Operators. B-6
B.2.3 Procedure Operators. B-6
B.2.4 Transfer Operators B-6

B.3 Counting Halstead Operands. B-11
B.4 Counting Halstead Input/Output Parameters B-12
B.5 Counting Halstead Operators and Operands:
An Example. B-12

Appendix C - McCabe's Measure in SAP C-1

C.1 Cyclomatic Complexity C-1
C.2 SAP Reports of the Cyclomatic Complexity. C-2
C.3 Calculation of the Cyclomatic Complexity. C-2

C.3.1 Decision Count From Computed GOTO Statements C-3
C.3.2 Decision Count From IF Statements. . . . C-3
C.3.3 Decision Count From Looping Statements . C-3
C.3.4 Decision Count Adjustment. C-4

Appendix D - User Complexity Techniques. D-1

D.1 Introduction. D-1
D.2 User's Statistical Weights File D-1
D.3 User Complexity Stubs D-3

TABLE OF CONTENTS (Cont'd)

Appendix E - Quality Index E-1

Appendix F - SAP Error Messages. F-1

References

Standard Bibliography of SEL Literature

LIST OF ILLUSTRATIONS

Figure

2-1	Sample NAMELIST	2-8
2-2	Sample JCL To Execute the Batch Version of SAP on the IBM 4341	2-10
2-3	Module Directory.	2-12
2-4	Global Summary.	2-15
2-5	Module Statistics Page.	2-16
2-6	Operand/Operator Summary.	2-17
2-7	Sample Sequential Output File	2-19
3-1	Keywords File	3-27
3-2	Default Statistical Weights File.	3-29
B-1	Sample Source Code.	B-16
B-2	Line-by-Line Summary of Halstead Counts	B-17
B-3	SAP Operand/Operator Summary Report	B-18
C-1	Control Graphs and Cyclomatic Numbers for Compound Decisions.	C-5
D-1	Sample User Statistical Weights File.	D-2

LIST OF TABLES

Table

2-1	SAP Listing Switches.	2-4
2-2	SAP External File Use Switches.	2-5
3-1	Module Summary External Communications Paragraph	3-3
3-2	External Communications Statistics Locator.	3-4
3-3	Module Summary Commenting Paragraph	3-5
3-4	Comment Statistics Locator.	3-6
3-5	Module Summary Statement Class Counter Paragraph	3-7
3-6	Statement Class Counter Locator	3-8
3-7	Module Summary Statement Type Paragraph	3-9
3-8	Statement Type Counter Locator.	3-11
3-9	Module Summary Control Statement Paragraph.	3-15
3-10	Control Statement Locator	3-16
3-11	Module Summary Assignment Statement Paragraph	3-17
3-12	Assignment Statement Statistics Locator	3-18
3-13	Module Summary Specification Statement Breakdown Paragraph	3-19
3-14	Specification Statement Statistics Locator.	3-20
3-15	Module Summary Complexity Analysis Paragraph.	3-21
3-16	Complexity Analysis Statistic Locator	3-22

LIST OF TABLES (Cont'd)

Table

3-17	Miscellaneous Statistics.	3-23
3-18	Miscellaneous Statistics Locator.	3-24
3-19	Statement Class Definitions	3-28
A-1	Assignment/Control Statement Type Summary . . .	A-2
A-2	Input/Output Statement Type Summary	A-3
A-3	Specification/Typing Statement Type Summary . .	A-4
A-4	Subprogram and Other Statement Type Summary . .	A-5
B-1	Delimiter Operators	B-3
B-2	Statement Types Examined for Delimiter Operators	B-4
B-3	Keyword Operators	B-7
B-4	Statement Types Examined for Procedure Operators	B-9
B-5	Transfer Operators.	B-10
B-6	Statement Types Examined for Operands	B-13

SECTION 1 - INTRODUCTION

The FORTRAN Static Source Code Analyzer Program (SAP) automatically produces statistics on occurrences of statements and structures within FORTRAN program modules and provides a facility for reporting these statistics. SAP is available in versions to run on either a PDP-11/70, VAX-11/780, or IBM 4341 computer. This document is a revision of the previous SAP user's guide, SEL-78-102, which describes SAP Version 2, which runs on the PDP-11/70 computer. This document describes Version 3, which runs on the VAX-11/780 and IBM 4341 computers.

SAP accepts as input syntactically correct FORTRAN source code written in the FORTRAN 77 standard language (Reference 1). In addition, code written using features in the following languages is also accepted: PDP-11 FORTRAN IV or FORTRAN IV-PLUS (References 2 and 3); VAX-11 FORTRAN (References 4 and 5); IBM VS FORTRAN (References 6 and 7); and Structured FORTRAN (Reference 8).

SAP operates in an interactive environment on the VAX-11/780 in which the user is prompted for a file name to specify the source code to be analyzed. The file name may be modified by optional control switches, which are used to specify the types of processing to be performed and the specific output files to be created. The IBM version runs in a batch environment.

The program uses two external permanent files: a keywords file and a statistical weights file. The keywords file provides flexibility in classifying statements as executable or nonexecutable. The statistical weights file is used in the computation of the Software Engineering Laboratory (SEL) complexity (Section 3.4). The user may specify a different

set of weights and thus redefine the composition of this measure of complexity.

SAP can produce three types of output: reports formatted for the line printer, files containing statistical results in a format readable by other analysis programs, and error and warning messages directed to the user's terminal.

This document is a detailed user's guide for SAP. Section 2 presents the instructions for operating the SAP program. The information presented describes starting the program, controlling the source code processing, stopping the program, and examining SAP output. Section 3 presents an overview of the SAP system, describing the content of SAP output files and presenting a simplified description of how source code is processed.

Five appendixes are provided to describe detailed information for researchers interested in how SAP collects and calculates statistics. Appendix A describes how each FORTRAN statement type is processed. Appendix B describes how Halstead's measures (Reference 9) are gathered. Appendix C describes how McCabe's measure (Reference 10) and the number of decisions are counted. Appendix D describes how the statistical weights file and the user complexity "stubs" (UCPLX1 and UCPLX2) are used to calculate the user's own definition of complexity for a module. Appendix E describes the computation of the software quality index. An additional appendix, Appendix F, presents the error messages produced by SAP, along with an explanation of the probable cause of each error.

SECTION 2 - SAP PROGRAM OPERATION

The SAP program, as an interactive program, is operational on the VAX-11/780. Section 2.2 describes the use of SAP on this machine. Section 2.3 describes how to run SAP as a batch job on the IBM 4341 computer. This section is divided into four parts: a description of SAP input files and how to prepare them for processing, a description of the user's interactive control of SAP, a description of the user's batch control of SAP, and a description of SAP output files.

2.1 SAP INPUT FILES

The input to SAP is a file in American Standard Code for Information Interchange (ASCII) or Extended Binary Coded Decimal (EBCDIC) containing syntactically correct FORTRAN source code. SAP accepts code written in the FORTRAN 77 standard language. In addition, code written using the features in the following languages is also accepted: PDP-11 FORTRAN IV or FORTRAN IV-PLUS (References 2 and 3); VAX-11 FORTRAN (References 4 and 5); IBM VS FORTRAN (References 6 and 7); and structured FORTRAN (Reference 8).

Source code brought from non-DEC computers should be on tape in a fixed-block, fixed-record-length format. The International Business Machines Corporation (IBM) utility IEBGENER (Reference 11) or a suitable update program (for example, PACKUPD or PANVALET) may be used to create sequential tapes in this format.

2.1.1 READING FOREIGN TAPES ON THE VAX-11/780

On the VAX-11/780, the local utility, TAPECOPY, is used to move and translate foreign tape files to Files-11 formatted disk files. TAPECOPY assumes that each physical record on the tape is composed of multiple 80-byte card images. Trailing blanks are removed from each card image before being written to disk. To run TAPECOPY, the user enters the

following commands (user input is underlined and <CR> indicates a carriage return):

```
$ ALLOCATE MTA0: <CR>
$ MOUNT/FOREIGN/DENSITY=1600 MTA0: - <CR>
$_ LABEL LOGNAM <CR>
$ RUN DBB1:[FDYN]TAPECOPY <CR>
```

TAPECOPY will prompt the user for the tape file number, EBCDIC-to-ASCII translation, and destination file name.

2.1.2 READING FOREIGN TAPES ON THE IBM 4341

Source code brought to the IBM 4341 computer must be in a known, standard, labeled format. The IBM utility IEBGENER (Reference 11) can be used to transfer data from the tape to disk. The following job control language (JCL) sample will transfer data from the tape TAPE to the temporary data set USERID.SOURCE.FORT. The source code was written to the tape with a fixed, blocked record format; a record size of 80, and a block size of 2000.

```
//S1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN    DD DUMMY
//SYSUT1   DD DSN=TAPE,UNIT=TAPE,VOL=SER=TAPE,LABEL=
//            (1,BLP), DCB=(DEN=3,RECFM=FB,LRECL=80,
//            BLKSIZE=2000),DISP=(OLD,PASS)
//SYSUT2   DD DSN=USERID.SOURCE.FORT,UNIT=DISK,DISP=
//            (NEW,CATLG),DCB=(RECFM=FB,LRECL=80,
//            BLKSIZE=2000,DSORG=PO),SPACE=(TRK,
//            (10,5,10))
```

This JCL assumes that the source file is the first file on the tape because of the 1 in the label field. Reference 11 contains a complete discussion of IEBGENER, and Reference 12, a discussion of JCL.

2.2 SAP INTERACTIVE EXECUTION ON THE VAX-11/780

SAP is executed interactively on the VAX-11/780 computer as follows:

```
(VAX)      $ RUN DBB1:[TOOLS]SAP
```

After the user has entered the appropriate RUN command shown above, the program will respond with the following prompt:

```
SAP>
```

The user specifies and controls SAP processing in this stage by entering the names of files to be analyzed and appending control switches as required. To exit the SAP program, the user enters a control Z (^Z) in response to the SAP> prompt.

The following general format is used to specify a source code input file and processing options:

```
SAP> fileinl/S1/S2.../SN <CR>
SAP>
```

where fileinl is the source input file specifier and /S1 through /SN are control switches. The file specifier may include a device and directory name if the user wishes to refer to a device or directory that is not the current default. The control switches are used to control SAP listing output and to direct SAP to use some external files in its processing. Tables 2-1 and 2-2 list the legal control switches.

SAP control switches are used to specify output listing contents and the use of external files. The switches are turned on (/XX) and off (/-XX) by appending the switch to the end of a file specification. Once a switch setting has been specified, the switch remains at that setting until respecified or until the end of a run.

In the following sequence, the user specifies that file DOTEST.FOR is to be processed with the /MO and /GB switches on and all other switches set to off (the default), file IFTEST.FOR is processed with no module statistics output,

Table 2-1. SAP Listing Switches

Switch	Default	Description
/MO	/MO	Print entire module statistics summary
/EC	/-EC	Print module external communications paragraph (if /-MO in effect)
/CO	/-CO	Print module commenting paragraph (if /-MO in effect)
/SC	/-SC	Print module statement class counter paragraph (if /-MO in effect)
/ST	/-ST	Print module statement type counter paragraph (if /-MO in effect)
/CS	/-CS	Print module control statement breakdown paragraph (if /-MO in effect)
/AS	/-AS	Print module assignment statement breakdown paragraph (if /-MO in effect)
/SP	/-SP	Print module specification statement breakdown paragraph (if /-MO in effect)
/CA	/-CA	Print module complexity analysis paragraph (if /-MO in effect)
/HL	/-HL	Print module operator/operand summary paragraphs
/GB	/GB	Print global summary of statistics for input file
/LI	/-LI	Print source code listing
/DU	/-DU	Print formatted dump of the symbol table

and file SHORTEST.FOR is to be processed with INCLUDE statements expanded and no module statistics output.

```
RUN SAP
SAP> DOTEST.FOR
SAP> IFTEST.FOR/-MO
SAP> SHORTEST.FOR/XP
  :
```

Table 2-2. SAP External File Use Switches

Switch	Default	Description
/XP	/-XP	Expand (DEC) INCLUDE statements before processing
/UW	/-UW	Use an alternate file for SEL complexity weights
/SL	/-SL	Write module statistics to SAP sequential output file, ALL.SAP

The following should be noted when specifying SAP control switches:

- /MO Control Switch. When turned on, this switch specifies that the entire module statistics page (Section 2.4.2) is to be produced for each module regardless of the settings for the module paragraph switches (/EC, /CO, /SC, /ST, /CS, /AS, /SP, and /CA). When the /MO switch is turned off (/-MO), only the paragraphs with the corresponding switch set to on will appear on the module statistics page.

- /HL Control Switch. This switch controls the module operator/operand listing. This report always starts at the top of a separate page following the module statistics page (Section 2.4.2) if one is produced. This listing is not considered part of the module summary and so is not affected by the setting of the /MO switch.

- /GB Control Switch. This switch controls the global summary statistics report (Section 2.4.1). A module directory and a global statistics report will appear for each source input file while the /GB switch is on.

- /XP Control Switch. This switch instructs SAP to expand all INCLUDE statements (PDP and VAX FORTRAN only) encountered. INCLUDE statements will be expanded to a nesting depth of three. PANVALET ++INCLUDE statements cannot be expanded. Failure to expand INCLUDE statements may result in misleading statistical values.

This switch may be used to combine many source code files into one file. The user creates a small file consisting only of INCLUDE statements that refer to the files to be combined. When this file is processed by SAP with the /XP control switch set to on, SAP processes the files named on the INCLUDE statements as if they were one large sequential file. This procedure is a flexible and efficient way to repeatedly process the source code for a complete system. The advantages of this technique are that source code files can be added or removed easily by changing only one line of the file, the most recent version of the file is always used, and there is only one extra file to maintain.

- /UW Control Switch. This switch instructs SAP to use an alternate statistical weights file as the source of weights used in computing the SEL complexity (Section 3.4). After using the /UW switch, specifying /-UW will cause SAP to revert to the default statistical weights file.

- /SL Control Switch. This switch instructs SAP to write statistical data to a sequential output file (Section 2.4.4) for possible use by other analysis programs. When this switch is specified, a file, ALL.SAP, is either created or opened for extension in the user's directory.

After locating or creating ALL.SAP, SAP will prompt the user for a project name up to eight characters in length, and a two-character subsystem prefix. These identifiers will be included on each record written to ALL.SAP during the session.

2.3 SAP BATCH EXECUTION ON THE IBM 4341

The IBM 4341 computer to which SAP was moved for this present release has an MVS operating system, hence it does not easily support interactive programs. All other capabilities of SAP were retained.

Command input to SAP is through a data set containing a string of variable names and values called a NAMELIST. Figure 2-1 is a sample NAMELIST. The NAMELIST accepts any switch in Tables 2-1 or 2-2 (except the /XP or /UW switches) minus the preceding slash. The switches can be input in any order. The switches are set to be equal to T for true and F for false, which is equivalent to typing the switch or its negative in the interactive version (Section 2.2). The two other variables in the NAMELIST are PROJNM and PREFIX. These are character variables that are used when the /SL switch is set to true to write to the sequential output file (Section 2.4.4). PROJNM is a project name up to eight characters long. PREFIX is the project prefix up to two characters long. If a variable is not specified in the NAMELIST, the same default value in Tables 2-1 and 2-2 is used for the switch. Not specifying a value for PROJNM or PREFIX will result in using the previous value, or blanks if no previous value was specified. Specifying an incorrect value for a NAMELIST variable will cause a NAMELIST read error. That variable will be ignored, and execution will continue.

Certain rules must be satisfied when writing the NAMELIST. The NAMELIST name "&SWITCH" must start in column 2 and be followed by a blank for every data group. The variables and

values follow this, separated by commas. The end of a data group must be an "&END". The NAMELIST must not contain any line numbers. PROJNM must be padded with blanks to the right so that it is exactly eight characters long. PREFIX must be padded with blanks to the right so that it is exactly two characters long. Comments within the NAMELIST are ignored. Every data group corresponds to one response from a SAP> prompt in the interactive version. There is no limit on the number of data groups in a NAMELIST as long as there are enough source files for the program to read. This will be explained in the discussion of setting up the JCL deck. Reference 7 contains more information on NAMELISTs.

```
DATE: 10:13:04- 3/27/85 NAMELIST.SAP;2          CREATE:22-MAR-1985 PAGE:   1
                                                  USRDISK:<TOOLS.T0005.SAP>

    1. C  SAP NAMELIST
    2. C
    3. C
    4.  &SWITCH
    5.    MO=T,HL=T,GB=T,LI=T,SL=T,PROJNM='TEST    ',PREFIX='SA',
    6.  &END
    7. C
    8.  &SWITCH
    9.   MO=F,CA=T,GB=T,LI=T,SL=T,PROJNM='SECOND  ',
   10.  &END
```

Figure 2-1. Sample NAMELIST

The /UW switch is not used, because the user specifies the weights file in the JCL deck. The batch version of SAP cannot expand the include statements while it is running. To process an expanded version of the source, it must be expanded previously and written to a data set. This data set name is used in the JCL to run the batch version.

Figure 2-2 is a sample of a JCL deck used to execute the batch version. The program is initiated by this JCL submission to the operating system. The NAMELIST must have been set up previously, and all the data sets referenced must

```
DATE: 10:13:12- 3/27/85 JCL.SAP;2                CREATE:26-MAR-1985 PAGE:    1
                                                    USRDISK:<TOOLS.TOOO5.SAP>

  1. //*
  2. //*       RETIREVE KEYWORD FILE
  3. //*
  4. //KEY EXEC  PANUPDT,REGION=120K,
  5. //          LIB='GJFEM.SAP.PAN'
  6. //PANDD2 DD DISP=(NEW,PASS),DSN=&&KEYWRD,
  7. //          UNIT=DISK,SPACE=(TRK,(5,1),RLSE),
  8. //          DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
  9. //SYSIN DD *
 10. ++WRITE WORK,DAKEYWRD
 11. //*
 12. //*       RETRIEVE WEIGHT FILE
 13. //*
 14. //WGHT EXEC PANUPDT,REGION=120K,
 15. //          LIB='GJFEM.SAP.PAN'
 16. //PANDD2 DD DISP=(NEW,PASS),DSN=&&WEIGHTS,
 17. //          UNIT=DISK,SPACE=(TRK,(5,1),RLSE),
 18. //          DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
 19. //SYSIN  DD *
 20. ++WRITE WORK,DAWEIGHT
 21. //*
 22. //*       GO
 23. //*
 24. //GO EXEC PGM=SAP,REGION=700K
 25. //STEPLIB  DD DSN=GJFEM.SAP.LOAD,DISP=SHR
 26. //FT06F001 DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)
 27. //FT20F001 DD DSN=&&KEYWRD,DISP=(OLD,DELETE)
 28. //FT21F001 DD DSN=source file to be processed, DISP=SHR
 29. //FT21F002 DD DSN=source file to be processed, DISP=SHR
 30. //FT22F001 DD DSN=&&WEIGTHS,DISP=(OLD,DELETE)
 31. //FT24F001 DD DSN=user's namelist, DISP=SHR
 32. //FT25F001 DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)
 33. //FT26F001 DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)
 34. //FT27F001 DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)
 35. //FT31F001 DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)
 36. //FT15F001 DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1922)
 37. /
```

Figure 2-2. Sample JCL To Execute the Batch Version of SAP on the IBM 4341

exist and be accessible. Input files are connected to units 20 through 24. Units 20 and 22 are the keywords and weights data files, which are retrieved from the PANVALET library GJFEM.SAP.PAN in lines 10 through 200. These two files are written to temporary sequential files, which are deleted at the end of execution. The name of an alternate weights data file should be specified in line 300 instead of using the /UW switch. Unit 21 in lines 280 through 290 has the name of the input source file that SAP will process. Unit 24 has the name of the data file that contains the NAMELIST. Multiple source files can be processed by observing the steps in the FTnnF00n keyword for unit 21 in lines 280 through 290. The executable load module is accessed in lines 240 through 250.

Units 25 through 31 are the output files. Unit 25 is used for the listing file (Section 2.4.3). Unit 26 is used for the module statistics file (Section 2.4.2). Unit 27 is used for the global statistics file (Section 2.4.1). Unit 31 is used for the sequential output file. In this JCL example, all output is written to the printer. To write the output to a file, the SYSOUT and DCB portion of the command line should be changed as follows:

```
//FTnnF00n DD DSN=users choice,UNIT=DISK,SPACE=(TRK,
//            (50,10)),DISP=(NEW,CATLG),DCB=(RECFM=FB,
//            LRECL=80,BLKSIZE=2000)
```

This will create a temporary data set with the user's choice of name. To permanently catalog the data set, the user should specify the volume name. The space and DCB parameters can be changed to suit the user's needs. Reference 12 contains more information on the JCL.

2.4 SAP OUTPUT

SAP output consists of three listing files and one data file. The listing files and their contents are as follows:

```
FOR008.DAT - Module directory (/GB switch)
           - Global summary (/GB switch)
FOR007.DAT - Module statistics (/MO switch)
           - Operator/operand summary (/HL switch)
FOR006.DAT - Source code listing (/LI switch)
           - Symbol table dump (/DU switch)
           - Error and warning messages
```

These files are described in Sections 2.4.1 through 2.4.3.

The SAP output data file is a sequential output file discussed in Section 2.1 under the /SL control switch. The contents of this file are described in Section 2.4.4.

For detailed information on all the statistics that appear in the listing files and the output data files, see Section 3.2. That section shows or references the method of calculation and lists the location of each statistic within the reports and files.

2.4.1 GLOBAL STATISTICS FILE (FOR008.DAT)

The global statistics file contains the listings defined by the /GB switch. A new version of this file is created for each session with SAP in which output is directed to this file. The module directory and the global summary are described below.

2.4.1.1 Module Directory

Figure 2-3 is an example of the module directory. The output contains information on a module-by-module basis, as follows:

- Module counter
- Module name

1 2 SOURCE ANALYZER PROGRAM V3 .GLOBAL STATISTICS SUMMARY FILE LUN 8 ADDPOT PAGE 1

SAMPLE.FOR/HL/SL/XP

MODULE DIRECTORY

	MODULE NAME		SOURCE LINES	SOURCE CMTS	EXEC STMT	N-EXC STMTS	STATMENT INFORMATION COUNT ASGN	I/O	CNTL	STRU	INCL	OTHER	HALSTEAD OPTR	HALSTEAD OPND	****** CYCLO	COMPLEXITITES SEL	****** USER1	****** USER2	PAGE NO.	NUMBER OF ERRS	NUMBER OF WARN
1	ADDPOT	SU	90	69	9	9	5	1	3	2	0	0	36	50	2	90.5	0.0	0.0	1	0	0
2	ASGNID	SU	162	88	35	31	10	0	25	12	0	2	161	230	12	290.0	0.0	0.0	3	0	0
3	CINPUT	SU	225	98	78	49	40	4	34	22	0	5	335	277	20	632.5	0.0	0.0	5	0	0
4	CNTXFR	SU	140	112	9	12	5	0	4	2	0	2	38	96	3	120.5	0.0	0.0	7	0	0
5	COLGLB	SU	338	203	111	29	77	0	34	0	0	1	525	715	34	1053.5	0.0	0.0	9	0	0
6	COMPAR	SU	50	32	10	9	5	0	5	4	0	1	37	24	5	70.0	0.0	0.0	11	0	0
7	COMPQI	SU	327	219	60	34	41	0	19	12	0	0	320	458	19	699.0	0.0	0.0	13	0	0
8	COMPWT	SU	264	173	55	24	49	0	6	0	0	1	236	348	6	675.0	0.0	0.0	15	0	0
9	CT1BLK	BL	31	23	0	6	0	0	0	0	0	0	6	22	1	24.0	0.0	0.0	17	0	0
10	CT2BLK	BL	40	31	0	6	0	0	0	0	0	0	6	30	1	29.0	0.0	0.0	19	0	0
11	CT3BLK	BL	19	12	0	6	0	0	0	0	0	0	7	13	1	18.5	0.0	0.0	21	0	0
12	CT4BLK	BL	13	6	0	7	0	0	0	0	0	0	11	12	1	16.0	0.0	0.0	23	0	0
13	CT5BLK	BL	68	57	0	6	0	0	0	0	0	0	7	54	1	43.0	0.0	0.0	25	0	0
14	DEFSEL	SU	99	54	21	20	7	6	8	2	0	2	73	65	7	121.0	0.0	0.0	27	0	0
15	DELBLK	BL	17	7	0	6	0	0	0	0	0	0	65	65	1	33.0	0.0	0.0	29	0	0
16	DLIBLK	BL	24	8	0	6	0	0	0	0	0	0	38	37	1	17.5	0.0	0.0	31	0	0
17	DSCAN	SU	191	93	65	35	38	0	27	19	0	4	224	184	18	517.0	0.0	0.0	33	0	0
18	ERAPOT	SU	56	49	3	3	2	0	1	0	0	0	9	19	1	33.5	0.0	0.0	35	0	0
19	ERRMSG	SU	143	89	20	32	3	5	12	9	0	1	108	104	8	106.5	0.0	0.0	37	0	0
20	ESTIM	SU	130	82	36	7	34	0	2	2	0	0	127	110	3	398.0	0.0	0.0	39	0	0
21	FINDIT	SU	71	47	14	10	10	0	4	4	0	1	54	41	4	126.0	0.0	0.0	41	0	0
22	FLVARI	SU	229	161	43	21	21	0	22	4	0	1	146	227	15	361.0	0.0	0.0	43	0	0
23	FNNAME	SU	70	43	15	12	10	0	5	5	0	1	66	49	5	129.0	0.0	0.0	45	0	0
24	GARCOL	SU	133	86	29	20	16	1	12	7	0	3	116	111	10	223.0	0.0	0.0	47	0	0
25	GLBBLK	BL	36	19	0	11	0	0	0	0	0	0	45	82	1	38.0	0.0	0.0	49	0	0
26	GLINE	SU	257	129	84	41	43	6	35	16	0	6	377	325	26	637.0	0.0	0.0	51	0	0
27	TDIST	SU	51	13	23	14	15	1	7	2	0	3	83	67	5	198.0	0.0	0.0	53	0	0

Figure 2-3. Module Directory

- Two-letter descriptor of module type (BL (BLOCK DATA), MA (Main Program), FU (FUNCTION) and SU (SUBROUTINE))
- Number of source lines of code
- Number of comment lines
- Number of executable statements
- Number of nonexecutable statements
- Number of assignment statements
- Number of input/output statements
- Number of control statements
- Number of structure statements
- Number of INCLUDE statements
- Number of undecoded statements
- Total usage of all Halstead operators appearing in the module (Appendix B)
- Total usage of all Halstead operands appearing in the module (Appendix B)
- Cyclomatic complexity (Appendix C)
- SEL complexity (Appendix D)
- User's complexity number 1 (Appendix D)
- User's complexity number 2 (Appendix D)
- Page number for module statistics report in file FOR007.DAT
- Number of SAP errors (internal processing errors, such as table overflow)
- Number of SAP warnings (syntax errors in the module)

2.4.1.2 Global Summary

Figure 2-4 is an example of the global summary. The statistics are averages, sums, and/or maxima of the majority of the statistics appearing on the module statistics page (Section 2.4.2). For ease of comparison, the global summary page format is approximately the same as the module statistics page, although some statistics gathered for individual modules are not summarized on the global summary page. The global summary presents statistics for one input file.

2.4.2 MODULE STATISTICS FILE (FOR007.DAT)

Module output consists of the individual module statistics pages and the operator/operand summary pages. Figure 2-5 is an example of a module statistics page. The module statistics page is divided into eight "paragraphs". Each paragraph is labeled with a name appearing to the left (except for the fourth paragraph, Statement Type Counters, which is not labeled). The output of this report (and the paragraphs that appear in it) is controlled by the /MO switch and the auxiliary paragraph switches (Section 2.1, Table 2-1).

Figure 2-6 is an example of the operator/operand summary controlled by the /HL switch. The counts of the four types of operators (Appendix B) are presented in individual paragraphs. The summaries of delimiter and keyword operators list the counts of every possible operator. A zero count indicates the operator was not detected in the module. The procedure and transfer operators are listed only for those detected.

2.4.3 LISTING FILE (FOR006.DAT)

The listing file contains an echo of the input FORTRAN source code when the /LI switch is set. This file will also contain the symbol table dump produced when the /DU switch

SAMPLE.FOR/HL/SL/XP

***** GLOBAL SUMMARY *****

GLOBAL MODULE	TOTAL MODULES	NUMBER MAINS	NUMBER SUBROUTINES	NUMBER FUNCTIONS	NUMBER BLOCKDATAS	SOURCE LINES	CODE LINES	COMMENT LINES
	27	0	19	0	8	3274	1271	2003

MODULE COMMENTING STATISTICS	LINES PER MODULE TOTAL	CODE	COMMENT	PROLOGUE LENGTH	COMMENT LINES EMBEDDED	BLANK
	338 MAX.	135 MAX.	219 MAX.	198 MAX.	40 MAX.	58 MAX.
	121.3 AVG.	47.1 AVG.	74.2 AVG.	61.9 AVG.	12.3 AVG.	22.1 AVG.

MODULE COMMUNICATION STATISTICS	ENTRY POINTS	SUBR. CALLS	FUNCT. REF.	EXTERNAL NAMES	EXTERNALLY DEFINED	ASF DEFINITIONS	ASF REFERENCES	ARG. LIST LNGTH IN REFERENCES
	0 TOTAL	29 TOTAL	6 TOTAL	DEFINED	REFERENCES	PER MODULE	PER MODULE	TO SUBR/FUNCT.
	PER MODULE	PER MODULE	PER MODULE					
	0 MAX	7 MAX	1 MAX	15 MAX	0 MAX	0 MAX	0 MAX	7 MAX
	0.0 AVG	1.1 AVG	0.2 AVG	2.0 AVG	0.0 AVG	0.0 AVG	0.0 AVG	2.3 AVG

	EXECUTABLE CNT.	PCT.	STATEMENT	NON-EXECUTABLE CNT.	PCT.	STATEMENT	MISCELLANEOUS CNT.	PCT.	STATEMENT
STATEMENT	720	60.7	EXECUTABLE	466	39.3	NON-EXECUTABLE	0	0.0	NAMELIST
CLASS	431	36.3	ASSIGNMENT	54	4.6	SUBPROGRAM	22	1.9	DATA
COUNTERS	265	22.3	CONTROL	105	8.9	SPECIFICATION	0	0.0	ASF DEFINED
	124	10.5	STRUCTURED	108	9.1	TYPE SPECIF.	19	1.6	FORMAT
	24	2.0	I/O				34	2.9	OTHER

0	ASF DEF.	431	ASSIGNMENT	0	ACCEPT	0	ASSIGN	0	AT	0	BACKSPACE
8	BLOCKDATA	0	BYTE	29	CALL	32	CHARACTER	0	CLOSE	68	COMMON
0	COMPLEX	34	CONTINUE	22	DATA	0	DEBUG	0	DECODE	0	DEFINEFILE
0	DELETE	10	DIMENSION	0	DISPLAY	0	DOUBLECOMP	0	DOUBLEPREC	0	DOWHILE
16	DO	0	EJECT	0	ELSEIF	55	ELSE	0	ENCODE	0	ENDDO
0	ENDDEBUG	0	ENDFILE	69	ENDIF	27	END	0	ENTRY	27	EQUIVALENC
0	EXTERNAL	0	FIND	19	FORMAT	0	FUNCTION	41	GOTO	0	.IF
147	IF	0	IMPLICIT	0	INCLUDE	0	INQUIRE	41	INTEGER	0	INTRINSIC
18	LOGICAL	0	NAMELIST	3	OPEN	0	PARAMETER	0	PAUSE	0	PRINT
0	PROGRAM	4	READ	17	REAL	32	RETURN	0	REWIND	0	REWRITE
0	SAVE	0	STOP	19	SUBROUTINE	0	THEN	0	TRACEOFF	0	TRACEON
0	TYPE	0	WAIT	17	WRITE	0	UNDECODED	0	UNLOCK	0	VIRTUAL

CONTROL STATEMENT BREAKDOWN	IF STMTS PER MODULE MAX.	AVG.	BLOCKIF NESTING MAX.	AVG.	GOTO STMTS PER MODULE MAX.	AVG.	DO STMTS PER MODULE MAX.	AVG.	DO LOOP NESTING DEPTH MAX.	AVG.	STMTS PER DO LOOP MAX.	AVE.
	30	5.4	6	2.2	8	1.5	5	0.6	1	1.0	19	4.4

ASSIGNMENT STATEMENT BREAKDOWN	VARIABLES PER ASSIGNMENT		OPERATORS PER STATEMENT		SUBSCRIPT COMPLEXITY	
	2.3 AVG.	7 MAX.	0.7 AVG.	9 MAX.	3 MAX.	1.1 AVG.

SPECIFICATION STATEMENT BREAKDOWN	VARIABLES NAMED PER MODULE MAX.	AVG.	VARIABLES REFERENCED PER MODULE EXEC. STMTS MAX.	AVG.	COMMONS MAX.	AVG.	EQUIVALENCED NAMES PER MODULE MAX.	AVG.	DIMENSIONS PER ARRAY MAX.	AVG.	CHARACTERS PER VARIABLE MAX.	AVG.
	240	53.1	58	15.1	222	44.7	10	2.1	2	1.1	8	5.6

COMPLEXITY ANALYSIS	HALSTEAD ANALYSIS OPERATORS TOTAL	UNIQUE	OPERANDS TOTAL	UNIQUE	LEVEL PROGRAM	LANGUAGE	SEL COMPLEXITY	CYCLOMATIC COMPLEXITY	QUALITY INDEX	PREDICTED PROGRAM LENGTH	PREDICTED EFFORT REQUIRED
TOTAL	3256	413	3815	1670	0.970	41.800	6700.00	211	1815.623	12773.0	2409179.3
MAX.	525	40	715	260	0.094	11.504	1053.50	34	94.175	2160.0	326472.2
MEAN	120.6	15.3	141.3	61.9	0.036	1.548	248.15	7.81	67.245	473.074	89228.859
STD. DEV.	131.3	9.2	161.2	67.1	0.024	2.251	273.54	8.62	15.900	562.058	114029.922

Figure 2-4. Global Summary

SAMPLE.FOR/HL/SL/XP

MODULE TYPE = SUBROUTINE MODULE NAME = TDIST

EXTERNAL COMMUNICATIONS

COMMUNICATIONS TO EXTERNAL

NAMES REFERENCED	TOTAL REFERENCES	ARGUMENT LIST LENGTH	COMMUNICATIONS FROM EXTERNAL
3 SUBPROGRAM	3 SUBPROGRAMS	TO SUBROUTINES AND	0 ENTRY POINTS
1 FUNCTIONS	1 FUNCTIONS	FUNCTIONS	4 INPUT ARGUMENTS
0 ASF DEF.	0 ASF DEF.	2 MAX.	0 ENTRY ARGUMENTS
0 EXT. NAMES	0 EXT. NAMES	1.3 AVG.	2 RETURNS
0 NAMED COMMONS			

MODULE COMMENTING

COMMENT LINES		COMMENT PACKETS	CODE PACKETS	NUMBER OF LINES
1 BLANK	4 PROLOGUE	8 COUNT	13 COUNT	51 TOTAL
0 IN-LINE	9 EMBEDDED	SIZE(LINES)	SIZE(LINES)	38 CODE
		1.1 AVG. 2 MAX.	2.9 AVG. 9 MAX.	13 COMMENT

STATEMENT CLASS COUNTERS

EXECUTABLE			NON-EXECUTABLE			MISCELLANEOUS		
CNT.	PCT.	STATEMENT	CNT.	PCT.	STATEMENT	CNT.	PCT.	STATEMENT
23	62.2	EXECUTABLE	14	37.8	NON-EXECUTABLE	0	0.0	NAMELIST
15	40.5	ASSIGNMENT	2	5.4	SUBPROGRAM	0	0.0	DATA
7	18.9	CONTROL	0	0.0	SPECIFICATION	0	0.0	ASF DEFINED
2	5.4	STRUCTURED	6	16.2	TYPE SPECIF.	1	2.7	FORMAT
1	2.7	I/O				0	0.0	INCLUDE
						3	8.1	OTHER

0 ASF DEF.	15 ASSIGNMENT	0 ACCEPT	0 ASSIGN	0 AT	0 BACKSPACE
0 BLOCKDATA	0 BYTE	3 CALL	0 CHARACTER	0 CLOSE	0 COMMON
0 COMPLEX	3 CONTINUE	0 DATA	0 DEBUG	0 DECODE	0 DEFINEFILE
0 DELETE	0 DIMENSION	0 DISPLAY	0 DOUBLECOMP	0 DOUBLEPREC	0 DOWHILE
1 DO	0 EJECT	0 ELSEIF	1 ELSE	0 ENCODE	0 ENDDO
0 ENDDEBUG	0 ENDFILE	1 ENDIF	1 END	0 ENTRY	0 EQUIVALENC
0 EXTERNAL	0 FIND	1 FORMAT	0 FUNCTION	0 GOTO	0 .IF
1 IF	0 IMPLICIT	0 INCLUDE	0 INQUIRE	2 INTEGER	0 INTRINSIC
1 LOGICAL	0 NAMELIST	0 OPEN	0 PARAMETER	0 PAUSE	0 PRINT
0 PROGRAM	0 READ	3 REAL	2 RETURN	0 REWIND	0 REWRITE
0 SAVE	0 STOP	1 SUBROUTINE	0 THEN	0 TRACEOFF	0 TRACEON
0 TYPE	0 WAIT	1 WRITE	0 UNDECODED	0 UNLOCK	0 VIRTUAL

CONTROL STATEMENTS

IF		GOTO	UNCOND. GOTO	MODULE CONTROL			I/O STMT	
1 TOTAL	NESTING	0 TOTAL	0 OBJECT OF IF	0 PAUSE	1 DO LOOPS		BRANCH	TOTAL
0 LOGICAL	DEPTH	0 ASSIGNED	0 UPWARD	2 RETURN	NESTING	STATEMENTS	1 ERR=	2 BRANCHES
0 ARITH.	1.0 AVE	0 COMPUTED	0 DOWNWARD	0 RETURN I	DEPTH	PER LOOP	0 END=	2 TARGET
1 BLOCK	1 MAX	0 UNCOND.		0 STOP	1.0 AVG.	19.0 AVG.		LABELS
0 ELSE IF					1 MAX.	19 MAX.		

ASSIGNMENT STATEMENT

VARIABLES PER ASSIGNMENT		OPERATORS PER STATEMENT		SUBSCRIPT COMPLEXITY	
2.3 AVG.	4 MAX.	0.5 AVG.	1 MAX.	1.0 AVG.	1 MAX.

SPECIFICATION STATEMENT BREAKDOWN

VARIABLE NAMES	REFERENCED VARIABLES	ARRAY DIMENSIONS	NUMBER OF CHARACTERS IN A VARIABLE NAME
16 IN MODULE	16 IN CODE	1.0 AVG. 1 MAX.	2.1 AVG. 6 MAX.
0 IN COMMON	0 IN COMMONS		
0 EQUIVALENCED			

COMPLEXITY ANALYSIS

HALSTEAD ANALYSIS							
TOTAL		UNIQUE	LEVEL	PREDICTED	SEL COMPLEXITY	CYCLOMATIC COMPLEXITY	QUALITY INDEX
83	OPERATORS	19	0.02 PROGRAM	167 PROGRAM LENGTH	198.00	5	86.492
67	OPERANDS	20	0.30 LANGUAGE	40525.9 EFFORT REQUIRED			

Figure 2-5. Module Statistics Page

SAMPLE.FOR/HL/SL/XP

HALSTEAD OPERATORS

DELIMITERS

0	//	0	**	3	*	0	/	2	+	3	-
15	=	9	(	15	,	0	&	0	.NE.	0	.LT.
0	.LE.	0	.EQ.	0	.GE.	0	.GT.	0	.AND.	0	.OR.
0	.XOR.	0	.EQV.	0	.NOT.	0	.NEQV.				

KEYWORDS

0	ASSIGNTO	0	COMMON	0	DATA	1	DO=,,	0	DOWHILE	0	IMPLD DO
1	ELSE	0	ELSE IF	0	ENTRY	23	EOS	0	EQUIVLNC	0	IF()
0	IF(),,	1	.IF()	0	PARAMETR	0	PAUSE	2	RETURN	0	SAVE
0	STOP	0	ACCEPT	0	BACKSPCE	0	CLOSE	0	DECODE	0	DEFNFILE
0	DELETE	0	ENCODE	0	ENDFILE	0	FIND	0	INQUIRE	0	OPEN
0	PRINT	0	READ	0	REWIND	0	REWRITE	0	TYPE	0	WAIT
1	WRITE	0	UNLOCK	1	ERR =	0	END =	0	IOSTAT =		

PROCEDURES: EXTERNAL FUNCTIONS AND SUBROUTINES

1	VERIFY	1	SQRT	1	ERRMSG	1	ABORT

TRANSFERS

1	ERR=	300
1	ALT.RET.	ERRMSG 400

HALSTEAD OPERANDS

2	1	8	I	3	K	2	N	4	R	3	X	3	Y	1	27
4	R2	3	X2	3	Y2	4	0.0	4	DX	4	DY	4	XL	4	YL
3	ERR	4	DIST	1	SQRT	3	MSGNUM								

Figure 3-6. Operand/Operator Summary

is set. The format and content of the symbol table dump are given in the SAP system description document (Reference 13).

SAP error and warning messages are directed to this file. Most SAP messages have the following format:

```
***** routine type ***** - msg
```

where routine is the name of the SAP module producing the message

type is the type of message (ERROR or WARNING)

msg is the narrative description of the error or warning

Appendix F lists all SAP error messages and discusses the most probable cause for each error.

2.4.4 SEQUENTIAL OUTPUT FILE

The sequential file, ALL.SAP, contains a single formatted record for each module processed while the /SL switch is set. The current content and format of this file are based on the requirements specified for transferring data from SAP to the SEL software development data base component information files (Reference 14). Figure 2-7 is a listing of a sample sequential file. The following information is contained on each record:

- Project name
- Subsystem prefix
- Module name
- Number of arguments passed to the module
- Number of blank comment lines
- Number of executable statements
- Number of input/output (ACCEPT, PRINT, READ, TYPE, and WRITE) statements
- Total number of source lines

```
DATE: 10:13:25- 3/27/85 ALL.SAP;1                    CREATE:27-MAR-1985 PAGE:   1
                                                       USRDISK:<TOOLS.TOOO5.SAP>

  1. SAMP    SUADDPOT  2 69   9 1  90 13 30  36  50  1  1  7  5 0  0  2  4  5  1  1
  2. SAMP    SUASGNID  3 88  35 0 162 22138 161 230 11 11 15 12 0  0 12 28 10  6  0
  3. SAMP    SUCINPUT  2 98  78 2 225 36 76 335 277 17 19 12 10 0  0 22 29 40  7  2
  4. SAMP    SUCNTXFR  0112   9 0 140 14 72  38  96  1  2  4  4 1  0  2  0  5  0  0
  5. SAMP    SUCOLGLB  0203 111 0 338 13260 525 715 30 33 56 56 3  0  0  0 77  0  0
  6. SAMP    SUCOMPAR  5 32  10 0  50 15 10  37  24  3  4  5  0 0  0  4  0  5  0  0
  7. SAMP    SUCOMPQI  1219  60 0 327 18260 320 458 18 18 24 23 0 15 12 15 41  0  0
  8. SAMP    SUCOMPWT  0173  55 0 264 10142 236 348  0  5 22 22 5  0  0  0 49  0  0
  9. SAMP    SUCT1BLK  0 23   0 0  31  5 19   6  22  0  0  1  1 0  0  0  0  0  0  0
 10. SAMP    SUCT2BLK  0 31   0 0  40  5 27   6  30  0  0  1  1 0  0  0  0  0  0  0
 11. SAMP    SUCT3BLK  0 12   0 0  19  6 10   7  13  0  0  1  1 0  0  0  0  0  0  0
 12. SAMP    SUCT4BLK  0  6   0 0  13  7  6  11  12  0  0  2  2 0  0  0  0  0  0  0
 13. SAMP    SUCT5BLK  0 57   0 0  68  6 51   7  54  0  0  1  1 0  0  0  0  0  0  0
 14. SAMP    SUDEFSEL  0 54  21 5  99 19 23  73  65  3  6  4  4 2  0  2  0  7  0  5
 15. SAMP    SUDELBLK  0  7   0 0  17  7 61  65  65  0  0  0  0 0  0  0  0  0  0  0
 16. SAMP    SUDLIBLK  0  8   0 0  24  6  9  38  37  0  0  2  2 0  0  0  0  0  0  0
 17. SAMP    SUDSCAN   1 93  65 0 191 28 41 224 184 15 17 14 13 0  1 19 11 38  3  0
 18. SAMP    SUERAPOT  0 49   3 0  56  6 15   9  19  0  0  3  3 0  0  0  0  2  0  0
 19. SAMP    SUERRMSG  3 89  20 5 143 21 45 108 104  7  7 14 11 0  0  9  2  3  1  5
 20. SAMP    SUESTIM  17 82  36 0 130 14 37 127 110  1  2 17  0 0  5  2  5 34  0  0
 21. SAMP    SUFINDIT  6 47  14 0  71 15 15  54  41  2  3  6  0 0  0  4  0 10  0  0
 22. SAMP    SUFLVARI  4161  43 0 229 25132 146 227 12 14 16 12 0  0  4  4 21  2  0
 23. SAMP    SUFNNAME  6 43  15 0  70 16 16  66  49  3  4  6  0 0  0  5  0 10  0  0
 24. SAMP    SUGARCOL  1 86  29 1 133 21 35 116 111  7  9  7  6 2  1  7  1 16  0  1
 25. SAMP    SUGLBBLK  0 19   0 0  36  6 45  45  82  0  0  3  3 0  0  0  0  0  0  0
 26. SAMP    SUGLINE   8129  84 6 257 40 75 377 325 15 25 16  8 2  2 16 22 43  6  4
 27. SAMP    SUTDIST   4 13  23 1  51 19 20  83  67  1  4  4  0 1  1  2  5 15  3  1
```

Figure 2-7. Sample Sequential Output File

- Count of unique Halstead operators
- Count of unique Halstead operands
- Total usage of Halstead operators
- Total usage of Halstead operands
- Count of IF (IF and .IF) statements
- Number of decisions
- Number of input/output parameters (module arguments, entry point arguments, and referenced COMMON block variables)
- Number of referenced COMMON block variables
- Count of DO statements
- Count of function references
- Count of structure statements
- Count of arguments to CALL statements
- Count of assignment statements
- Count of CALL statements
- Count of FORMAT statements

SECTION 3 - OVERVIEW OF SAP

3.1 SAP PROCESSING OVERVIEW

SAP statistics are gathered for individual modules, and overall statistics for a complete input file are accumulated for the global summary. For a given module, SAP processes one complete statement at a time as follows:

1. Read (and print if requested) a line of code, remove Hollerith and literal fields, compress remaining blanks, and append continuation cards
2. Perform an initial lexical analysis, identifying delimiters and separating the input line into tokens
3. Recognize assignment statements
4. Identify keyword statements
5. Perform specific statement type analysis and gather statistics
6. Process labels and identify the end of DO loop and block IF structures
7. If the current statement is a logical IF, repeat steps 3, 4, and 5 for the statement that is the object of the logical IF

In SAP, a token is defined as a string of one or more characters bounded by a predefined delimiter string or an end of line. All tokens in a module are entered into a symbol table using a chained hash access algorithm. This symbol table is initialized to empty before processing each module.

During statement processing, one of the following four events occurs:

1. Normal statistics on the statement are gathered.

2. The statement type is not identified and the statement is marked as UNDECODED.

3. The statement type is identified, but due to a syntax error, complete statistics on the statement are not gathered. This type of error may affect only the statistics for the single statement or may affect the validity of the overall statistics for the current module (that is, results are uncertain). If warnings occur, they are indicated by a count in the warning column (for that module) in the module directory (Section 2.4.1.1).

4. An internal SAP problem, such as symbol table overflow, occurs. In this case, the module processing, and possibly the entire input file processing, is terminated. If an internal SAP error occurs, it is indicated by a count in the error column (for that module) in the module directory. A symbol table dump can be used to determine the cause of the problem (Reference 13).

3.2 STATISTICS GATHERED

As stated, SAP gathers statistics on individual modules and on a global basis for an entire input file. The individual statistics are listed in Tables 3-1 through 3-18. The tables are grouped in pairs that correspond to the "paragraphs" on the Module Statistic page (Figure 2-5). For example, Tables 3-1 and 3-2 both describe the statistics appearing in the external communications paragraph of the module summary.

Table 3-1. Module Summary External Communications Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
NAMES REFERENCED	1	NUMBER OF SUBROUTINE NAMES DEFINED	NUMBER OF DIFFERENT SUBROUTINE NAMES APPEARING IN CALL STATEMENTS
	2	NUMBER OF FUNCTION NAMES REFERENCED	NUMBER OF DIFFERENT FUNCTION NAMES APPEARING IN MODULE
	3	NUMBER OF ARITHMETIC STATEMENT FUNCTIONS DEFINED	DIRECT COUNT
	4	NUMBER OF EXTERNAL NAMES DEFINED	DIRECT COUNT
	5	NUMBER OF COMMON BLOCKS REFERENCED	DIRECT COUNT
TOTAL REFERENCES	6	NUMBER OF SUBROUTINE CALLS	DIRECT COUNT
	7	NUMBER OF FUNCTION REFERENCES	DIRECT COUNT
	8	NUMBER OF REFERENCES TO ARITHMETIC STATEMENT FUNCTIONS	DIRECT COUNT
	9	NUMBER OF REFERENCES TO EXTERNALLY DEFINED NAMES	NUMBER OF REFERENCES TO SUBROUTINE AND FUNCTION NAMES PASSED TO MODULE THROUGH ARGUMENT LIST
ARGUMENT LIST LENGTH	10 11	LENGTH OF ARGUMENT LISTS IN REFERENCES TO SUBROUTINES AND FUNCTIONS: MAXIMUM AVERAGE	NUMBER OF ITEMS APPEARING IN ARGUMENT LISTS NOT COUNTING SUBSCRIPTS OF ARGUMENTS OR FUNCTIONS WITHIN AN ARGUMENT
FROM EXTERNAL	12	NUMBER OF ENTRY POINTS	DIRECT COUNT
	13	NUMBER OF ARGUMENTS TO MODULE	DIRECT COUNT
	14	NUMBER OF ARGUMENTS IN ALL ENTRY POINTS	COUNT NUMBER OF VARIABLES IN ENTRY ARGUMENT LISTS
	15	NUMBER OF RETURN STATEMENTS	DIRECT COUNT

8721/85

Table 3-2. External Communications Statistics Locator

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[a]	MODULE STATISTICS	SEQUENTIAL FILE	
1			●		47
2			●		44
3		mx A	●		56
4		mx A	●		53
5			●		49
6		T mx A	●	●	128
7		T mx A	●	●	45
8		mx A	●		55
9		mx A	●		56
10		mx	●		234
11		A	●		213
12		T mx A	●		48
13			●	●	244
14			●		236
15			●		178

8721-(91*)/85

[a]THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE. THE STATISTICS MAY BE REPORTED AS TOTALS (T), MAXIMA (mx), AND/OR AVERAGE (A).

Table 3-3. Module Summary Commenting Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
COMMENT LINES	16	NUMBER OF BLANK COMMENT LINES	DIRECT COUNT
	17	SUM OF (IN LINE) COMMENTS FOLLOWING AN ! ON THE SAME LINE AS A FORTRAN STATEMENT	DIRECT COUNT
	18	NUMBER OF COMMENTS IN PROLOG (COMMENTS APPEARING BEFORE FIRST EXECUTABLE STATEMENT IN MODULE)	DIRECT COUNT
	19	SUM OF ALL EMBEDDED (NONPROLOG) COMMENTS	DIRECT COUNT
COMMENT PACKETS	20	NUMBER OF COMMENT PACKETS	DIRECT COUNT
	21	LENGTH OF NONPROLOG COMMENT PACKETS: AVERAGE	NUMBER OF COMMENT LINES IN COMMENT PACKETS APPEARING AFTER FIRST EXECUTABLE STATEMENT
	22	MAXIMUM	
CODE PACKETS	23	NUMBER OF CODE PACKETS	DIRECT COUNT
	24	NUMBER OF LINES BETWEEN COMMENT PACKETS: AVERAGE	NUMBER OF CODE LINES BETWEEN COMMENT PACKETS
	25	MAXIMUM	
NUMBER OF LINES	26	TOTAL NUMBER OF SOURCE LINES	DIRECT COUNT
	27	NUMBER OF CODE LINES NOT INCLUDING IBM DEBUG	DIRECT COUNT
	28	NUMBER OF COMMENT LINES INCLUDING IBM DEBUG	DIRECT COUNT

8721/85

Table 3-4. Comment Statistics Locator

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[a]	MODULE STATISTICS	SEQUENTIAL FILE	
16		mx A	●	●	21
17			●		18
18		mx A	●		16
19		mx A	●		17
20			●		20
21			●		2
22			●		19
23			●		15
24			●		1
25			●		14
26	●	T mx A	●	●	11
27		T mx A	●		12
28	●	T mx A	●		13

8721-(91*1)/85

[a]THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE. THE STATISTICS MAY BE REPORTED AS TOTALS (T), MAXIMA (mx), AND/OR AVERAGE (A).

Table 3-5. Module Summary Statement Class Counter Paragraph

STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
	EXECUTABLE STATEMENTS:	THE METHOD OF CALCULATION FOR THE ENTRIES LISTED IN THIS TABLE IS BY DIRECT COUNT.
29	NUMBER	
30	PERCENTAGE	
	ASSIGNMENT STATEMENTS (CLASS 1):	
31	NUMBER	
32	PERCENTAGE	
33 34	NUMBER AND PERCENTAGE OF CONTROL STATEMENTS (CLASS 2)	
35 36	NUMBER AND PERCENTAGE OF STRUCTURED STATEMENTS (CLASS 10)	
37 38	NUMBER AND PERCENTAGE OF INPUT/OUTPUT STATEMENTS (CLASS 7)	
39 40	NUMBER AND PERCENTAGE OF NONEXECUTABLE STATEMENTS	
41 42	NUMBER AND PERCENTAGE OF SUBPROGRAM STATEMENTS (CLASS 3)	
43 44	NUMBER AND PERCENTAGE OF SPECIFICATION STATEMENTS (CLASS 4)	
45 46	NUMBER AND PERCENTAGE OF TYPE SPECIFICATION STATEMENTS (CLASS 5)	
47 48	NUMBER AND PERCENTAGE OF NAMELIST STATEMENTS (CLASS 9)	
49 50	NUMBER AND PERCENTAGE OF DATA STATEMENTS (CLASS 6)	
51 52	NUMBER AND PERCENTAGE OF DEFINED ARITHMETIC STATEMENT FUNCTIONS (CLASS 13)	
53 54	NUMBER AND PERCENTAGE OF FORMAT STATEMENTS (CLASS 8)	
55 56	NUMBER AND PERCENTAGE OF INCLUDE STATEMENTS (CLASS 11)	
57 58	NUMBER AND PERCENTAGE OF OTHER STATEMENTS (CLASS 12)	

8721/85

Table 3-6. Statement Class Counter Locator

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[a]	MODULE STATISTICS	SEQUENTIAL FILE	
29	●	T	●	●	73
30		T	●		71
31	●	T	●	●	101
32		T	●		81
33	●	T	●		102
34		T	●		82
35	●	T	●	●	110
36		T	●		90
37	●	T	●		107
38		T	●		87
39	●	T	●		74
40		T	●		72
41		T	●		103
42		T	●		83
43		T	●		104
44		T	●		84
45		T	●		105
46		T	●		85
47		T	●		109
48		T	●		89
49		T	●		106
50		T	●		86
51		T	●		99
52		T	●		85
53		T	●	●	108
54		T	●		88
55	●	T	●		111
56		T	●		91
57	●	T	●		112
58		T	●		92

8721-(91*1)/85

[a]THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE. THE STATISTICS MAY BE REPORTED AS TOTALS (T), MAXIMA (mx), AND/OR AVERAGE (A).

Table 3-7. Module Summary Statement Type Paragraph (1 of 2)

STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
59	ARITHMETIC STATEMENT FUNCTION DEFINITION	THE METHOD OF CALCULATION FOR THE ENTRIES LISTED IN THIS TABLE IS BY DIRECT COUNT.
60	ASSIGNMENT	
61	ACCEPT	
62	ASSIGN	
63	AT	
64	BACKSPACE	
65	BLOCKDATA	
66	BYTE	
67	CALL	
68	CHARACTER	
69	CLOSE	
70	COMMON	
71	COMPLEX	
72	CONTINUE	
73	DATA	
74	DEBUG	
75	DECODE	
76	DEFINEFILE	
77	DELETE	
78	DIMENSION	
79	DISPLAY	
80	DOUBLECOMPLEX	
81	DOUBLEPRECISION	
82	DOWHILE	
83	DO	
84	EJECT	
85	ELSEIF	
86	ELSE	
87	ENCODE	
88	ENDDEBUG	
89	ENDDO	
90	ENDFILE	
91	ENDIF	
92	END	
93	ENTRY	
94	EQUIVALENCE	
95	EXTERNAL	

8721-191*a)/85

Table 3-7. Module Summary Statement Type Paragraph (2 of 2)

STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
96	FIND	THE METHOD OF CALCULATION FOR THE ENTRIES LISTED IN THIS TABLE IS BY DIRECT COUNT
97	FORMAT	
98	FUNCTION	
99	GOTO	(INCLUDES COMPUTED, ASSIGNED, AND UNCONDITIONAL)
100	.IF	
101	IF	(INCLUDES ARITHMETIC, BLOCK, AND LOGICAL)
102	IMPLICIT	
103	INCLUDE	
104	INQUIRE	
105	INTEGER	
106	INTRINSIC	
107	LOGICAL	
108	NAMELIST	
109	OPEN	
110	PARAMETER	
111	PAUSE	
112	PRINT	
113	PROGRAM	
114	READ	
115	REAL	
116	RETURN	
117	REWIND	
118	REWRITE	
119	SAVE	
120	STOP	
121	SUBROUTINE	
122	THEN	
123	TRACEOFF	
124	TRACEON	
125	TYPE	
126	WAIT	
127	WRITE	
128	UNDECODED	
129	UNLOCK	
130	VIRTUAL	

8721-191*a)/85

Table 3-8. Statement Type Counter Locator (1 of 3)

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[a]	MODULE STATISTICS	SEQUENTIAL FILE	
59		T	●		121
60	●	T	●	●	122
61		T	●	b	123
62		T	●		124
63		T	●		125
64		T	●		126
65		T	●		127
66		T	●		128
67		T	●	●	129
68		T	●		130
69		T	●		131
70		T	●		132
71		T	●		133
72		T	●		134
73		T	●		135
74		T	●		136
75		T	●		137
76		T	●		138
77		T	●		139
78		T	●		140
79		T	●		141
80		T	●		142
81		T	●		143
82		T	●		144
83		T	●	●	145
84		T	●		146

8721-(91*a)/85

Table 3-8. Statement Type Counter Locator (2 of 3)

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[a]	MODULE STATISTICS	SEQUENTIAL FILE	
85		T	●		147
86		T	●		148
87		T	●		149
88		T	●		150
89		T	●		151
90		T	●		152
91		T	●		153
92		T	●		154
93		T	●		155
94		T	●		156
95		T	●		157
96		T	●		158
97		T	●	●	159
98		T	●		160
99		T	●		161
100		T	●	c	162
101		T	●	c	163
102		T	●		164
103	●	T	●		165
104		T	●		166
105		T	●		167
106		T	●		168
107		T	●		169
108		T	●		170
109		T	●		171
110		T	●		172

8721-191*a)/85

Table 3-8. Statement Type Counter Locator (3 of 3)

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[a]	MODULE STATISTICS	SEQUENTIAL FILE	
111		T	●		173
112		T	●	b	174
113		T	●		175
114		T	●	b	176
115		T	●		177
116		T	●		178
117		T	●		179
118		T	●	b	180
119		T	●	b	181
120		T	●		182
121		T	●		183
122		T	●		184
123		T	●		185
124		T	●		186
125		T	●	b	187
126		T	●		188
127		T	●	b	189
128	●	T	●		190
129		T	●		191
130		T	●		192

8721-(91*a)/85

[a]THE COUNTS ARE FOR THE TOTAL (T) FILE.

[b]THESE STATISTICS ARE SUMMED TO OBTAIN THE INPUT/OUTPUT STATEMENT COUNT.

[c]THESE STATISTICS ARE SUMMED TO OBTAIN THE IF STATEMENT COUNT.

Table 3-9. Module Summary Control Statement Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
IF	131	TOTAL NUMBER OF IF STATEMENTS	SUM OF LOGICAL, ARITHMETIC, BLOCK IF, AND ELSEIF STATEMENTS
	132	NUMBER OF LOGICAL IF STATEMENTS	DIRECT COUNT
	133	NUMBER OF ARITHMETIC IF STATEMENTS	DIRECT COUNT
	134	NUMBER OF BLOCK IF STATEMENTS	DIRECT COUNT OF BLOCK AND STRUCTURED IF
	135	NUMBER OF ELSEIF STATEMENTS	DIRECT COUNT
	136	AVERAGE NESTING DEPTH OF BLOCK IF STRUCTURES	SUM OF NESTED DEPTHS DIVIDED BY NUMBER OF BLOCK IF STATEMENTS
	137	MAXIMUM NESTING DEPTH OF IF STATEMENTS	DIRECT COUNT
GOTO	138	TOTAL NUMBER OF GOTO STATEMENTS	SUM OF ASSIGNED, COMPUTED, AND UNCONDITIONAL GOTO STATEMENTS
	139	NUMBER OF ASSIGNED GOTO STATEMENTS	DIRECT COUNT
	140	NUMBER OF COMPUTED GOTO STATEMENTS	DIRECT COUNT
	141	NUMBER OF UNCONDITIONAL GOTO STATEMENTS	DIRECT COUNT
UNCONDITIONAL GOTO	142	NUMBER OF GOTO STATEMENTS THAT ARE THE OBJECT OF A LOGICAL IF STATEMENT	DIRECT COUNT
	143	NUMBER OF UPWARD-POINTED UNCONDITIONAL GOTO STATEMENTS	COUNT OF UNCONDITIONAL GOTO STATEMENTS TO A LABEL OF A STATEMENT THAT HAS BEEN PREVIOUSLY SCANNED
	144	NUMBER OF DOWNWARD-POINTED UNCONDITIONAL GOTO STATEMENTS	COUNT OF UNCONDITIONAL GOTO STATEMENTS USING A LABEL NOT YET ENCOUNTERED
MODULE CONTROL	145	NUMBER OF PAUSE STATEMENTS	DIRECT COUNT
	146	NUMBER OF RETURN STATEMENTS	DIRECT COUNT (INCLUDES RETURN I)
	147	NUMBER OF RETURN I STATEMENTS	DIRECT COUNT
	148	NUMBER OF STOP STATEMENTS	DIRECT COUNT
	149	NUMBER OF DO STATEMENTS	DIRECT COUNT
	150	NESTING DEPTH OF DO LOOPS: AVERAGE	SUM OF NESTED DEPTHS DIVIDED BY NUMBER OF DO STATEMENTS
	151	MAXIMUM	DIRECT COUNT
	152	NUMBER OF STATEMENTS PER DO LOOP: AVERAGE	SUM OF NUMBER OF STATEMENTS PER DO LOOP DIVIDED BY NUMBER OF DO STATEMENTS
	153	MAXIMUM	DIRECT COUNT
INPUT/OUTPUT STATEMENTS	154	NUMBER OF ERR= KEYWORDS IN INPUT/OUTPUT STATEMENTS	DIRECT COUNT
	155	NUMBER OF END= KEYWORDS IN INPUT/OUTPUT STATEMENTS	DIRECT COUNT
MISCELLANEOUS	156	TOTAL NUMBER OF BRANCH STATEMENTS	COUNT OF IF AND GOTO STATEMENTS AND END= AND ERR= KEYWORDS
	157	TOTAL NUMBER OF TARGET LABELS	COUNT OF LABELS REFERENCED BY UNCONDITIONAL GOTO STATEMENTS AND ERR= AND END= KEYWORDS

8721-191-86

Table 3-10. Control Statement Locator

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[a]	MODULE STATISTICS	SEQUENTIAL FILE	
131		mx A	●		
132			●		221
133			●		222
134			●		255
135			●		256
136		A	●		
137		mx	●		258
138		mx A	●		
139			●		225
140			●		226
141			●		224
142			●		223
143			●		246
144			●		245
145			●		179
146			●		231
147			●		232
148			●		184
149		mx A	●	●	145
150		A	●		206
151		mx	●		248
152		A	●		207
153		mx	●		250
154			●		229
155			●		230
156			●		254
157			●		228

8721-(91*b)/85

[a]THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE. THE STATISTICS MAY BE REPORTED AS TOTALS (T), MAXIMA (mx), AND/OR AVERAGE (A).

Table 3-11. Module Summary Assignment Statement Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
VARIABLES PER ASSIGNMENT	158 159	NUMBER OF VARIABLES PER ASSIGNMENT: AVERAGE MAXIMUM	NUMBER OF VARIABLES IN ASSIGNMENT STATEMENTS
OPERATORS PER STATEMENT	160 161	NUMBER OF OPERATORS PER STATEMENT: AVERAGE MAXIMUM	NUMBER OF OPERATORS IN ASSIGNMENT STATEMENTS
SUBSCRIPT COMPLEXITY	162 163	SUBSCRIPT COMPLEXITY: AVERAGE MAXIMUM	NUMBER OF SUBSCRIPT PARENTHESIS PAIRS PLUS OPERATORS FOR ALL ARRAY SUBSCRIPTS IN MODULE

8721/85

Table 3-12. Assignment Statement Statistics Locator

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[a]	MODULE STATISTICS	SEQUENTIAL FILE	
158		A	●		204
159		mx	●		241
160		A	●		205
161		mx	●		243
162		A	●		208
163		mx	●		253

8721-(91*b)/85

[a]THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE. THE STATISTICS MAY BE REPORTED AS TOTALS (T), MAXIMA (mx), AND/OR AVERAGE (A).

Table 3-13. Module Summary Specification Statement Breakdown Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
VARIABLE NAMES	164	TOTAL NUMBER OF VARIABLES NAMED IN MODULE	DIRECT COUNT
	165	NUMBER OF VARIABLES NAMED IN COMMON BLOCK	DIRECT COUNT
	166	NUMBER OF VARIABLES EQUIVALENCED	NUMBER OF DISTINCT VARIABLE AND ARRAY NAMES THAT APPEAR IN EQUIVALENCE STATEMENTS
REFERENCED VARIABLES	167	NUMBER OF VARIABLES REFERENCED IN CODE	COUNT OF ALL VARIABLE AND ARRAY REFERENCES
	168	NUMBER OF COMMON BLOCK VARIABLES REFERENCED	NUMBER OF DISTINCT VARIABLE AND ARRAY NAMES (DEFINED IN COMMON) THAT APPEAR IN EXECUTABLE STATEMENTS
ARRAY DIMENSIONS	169 170	NUMBER OF DIMENSIONS FOR ARRAYS: AVERAGE MAXIMUM	DIRECT COUNT
NUMBER OF CHARACTERS	171 172	NUMBER OF CHARACTERS PER VARIABLE NAME: AVERAGE MAXIMUM	DIRECT COUNT

8721/85

Table 3-14. Specification Statement Statistics Locator

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[a]	MODULE STATISTICS	SEQUENTIAL FILE	
164		mx A	●		43
165			●		50
166		mx A	●		58
167		mx A	●		57
168		mx A	●		51
169		A	●		32
170		mx	●		60
171		A	●		31
172		mx	●		41

8721-(91*b)/85

[a]THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE. THE STATISTICS MAY BE REPORTED AS TOTALS (T), MAXIMA (mx), AND/OR AVERAGE (A).

Table 3-15. Module Summary Complexity Analysis Paragraph

	STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
BASIC HALSTEAD MEASURES	172	TOTAL NUMBER OF OPERATORS	SEE APPENDIX B
	173	TOTAL NUMBER OF OPERANDS	SEE APPENDIX B
	174	NUMBER OF UNIQUE OPERATORS	SEE APPENDIX B
	175	NUMBER OF UNIQUE OPERANDS	SEE APPENDIX B
DERIVED HALSTEAD QUANTITIES	176	PROGRAM LEVEL	SEE SECTION 3.5
	177	LANGUAGE LEVEL	SEE SECTION 3.5
	178	PREDICTED PROGRAM LENGTH	SEE SECTION 3.5
	179	PREDICTED EFFORT REQUIRED	SEE SECTION 3.5
OTHER COMPLEXITIES	180	SEL WEIGHTED COMPLEXITY	SEE SECTION 3.4, APPENDIX D
	181	CYCLOMATIC COMPLEXITY	SEE SECTION 3.6, APPENDIX C
	182	QUALITY INDEX	SEE APPENDIX E

8721-(91*c)/85

Table 3-16. Complexity Analysis Statistics Locator

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY[a]	MODULE STATISTICS	SEQUENTIAL FILE	
172	●	●	●	●	
173	●	●	●	●	
174		●	●	●	
175		●	●	●	
176		●	●		
177		●	●		
178		●	●		
179		●	●		
180	●	●	●		
181	●	●	●		
182		●	●		

8721-(91*c)/85

[a]THIS REPORT IS A SUMMARY FOR ALL MODULES IN A FILE. THE STATISTICS WILL BE REPORTED AS TOTALS (T), MAXIMA (mx), MEAN (A), AND STANDARD DEVIATION (SD).

Table 3-17. Miscellaneous Statistics (Not Reported on the Module Statistics Page)

STATISTIC NUMBER	ENTRY	METHOD OF CALCULATION
183	NUMBER OF DECISIONS	SEE APPENDIX C
184	NUMBER OF INPUT/OUTPUT PARAMETERS TO THE MODULE	SEE APPENDIX B
185	USER COMPLEXITY NUMBER 1	SEE APPENDIX D
186	USER COMPLEXITY NUMBER 2	SEE APPENDIX D
187	ACTUAL PROGRAM LENGTH	SEE SECTION 3.5
188	PROGRAM VOLUME	SEE SECTION 3.5
189	POTENTIAL PROGRAM VOLUME	SEE SECTION 3.5
190	PREDICTED TIME	SEE SECTION 3.5
191	PREDICTED BUGS	SEE SECTION 3.5

8721-191*c)/85

Table 3-18. Miscellaneous Statistics Locator

STATISTIC NUMBER	APPEARS IN				STATISTICAL WEIGHT INDEX
	MODULE DIRECTORY	GLOBAL SUMMARY	MODULE STATISTICS	SEQUENTIAL FILE	
183				●	
184				●	
185	●				
186	●				
187					
188					
189					
190					
191					

8721-191*c)/85

The first table in each pair describes the statistic and shows or references the method of calculation used to obtain the statistic. The first column in this table (Statistic Number) represents an arbitrary sequential ordering of all the statistics and is used only to provide a reference to the second table in the pair.

The second table in the pair shows the reports and data files in which the statistic appears. The reports and files listed are the module directory (Figure 2-3), the global summary (Figure 2-4), the project summary (Figure 2-5), the module statistics (Figure 2-5), and the sequential file (Figure 2-7). The last column shows the statistical weight index for the particular statistic (Section 3.4 and Appendix D).

As mentioned, the tables appear in pairs that correspond to the eight paragraphs on the Module Statistics page. The ninth pair of tables (Tables 3-17 and 3-18) lists statistics that do not appear on the Module Statistics page.

The tables refer to counts of lines and statements and to modules. The terms used in the tables are defined below.

- Line. A line is equivalent to a record in the input file. Comments are always counted in units of lines.

- Statement. A statement is composed of an initial line and any continuation lines that together contain a single FORTRAN statement. SAP makes one exception to this usual definition of a statement: a logical IF and the object statement of the logical IF are counted as separate statements (one each). Each statement type is classified as executable or nonexecutable. This classification is not the same as the classification used in the FORTRAN 77 standard and serves a different purpose than that classification. Tables A-1 through A-4 in Appendix A contain more information on statement classifications.

- Module. A module is the collection of statements preceding and including an END statement. Modules are classified as BLOCKDATA, SUBROUTINE, FUNCTION, or main program.

3.3 KEYWORDS FILE

To allow flexibility in classifying statements and in marking statements executable or nonexecutable, an external keywords file is used. The keywords file location on the VAX-11/780 is DBB1:[TOOLS]KEYWORDS.SAP, and in the PANVALET library GJFEM.SAP.PAN on the IBM 4341.

Figure 3-1 is a listing of the keywords file as it is currently implemented. Column 1 is a logical constant, which, if true(T), indicates the statement is executable, and if false(F), nonexecutable. Columns 2 and 3 are obsolete and do not affect the execution of SAP. Column 4 contains the statement class. Column 5 contains the number of characters in the keyword. Column 6 is the keyword. The format for a keywords file record is (L3, 6X, 2I3, 1X, 16A1).

Table 3-19 shows the definition of each statement class.

3.4 STATISTICAL WEIGHTS FILE

The statistical weights file is used in determining the SEL figure of complexity for each module. The formula used to determine this complexity is

$$\text{SEL complexity} = \sum_{i=1}^{256} (\text{module statistic})_i \times (\text{statistic weight})_i$$

The SEL complexity is reported in the module directory (Section 2.4.1.1) and on the module summary page (Section 2.4.2).

Most statistics printed on the module summary page have been assigned a statistical weight index as shown in Tables 3-1

```
DATE: 10:13:33- 3/27/85 KEYWORDS.SAP;1          CREATE:18-MAR-1985 PAGE:    1
                                                   USRDISK:<TOOLS.TOOO5.SAP>

 1.   F  O  O 11  8 ASF DEF.          37.   F  O  O  4  8 EXTERNAL
 2.   T  O  O  1 10 ASSIGNMENT        38.   T  O  O  7  4 FIND
 3.   T  O  O  7  6 ACCEPT            39.   F  O  O  8  6 FORMAT
 4.   T  O  O  2  6 ASSIGN            40.   F  O  O  3  8 FUNCTION
 5.   F  O  O 12  2 AT                41.   T  O  O  2  4 GOTO
 6.   T  O  O  7  9 BACKSPACE         42.   T  O  O 10  3 .IF
 7.   F  O  O  3  9 BLOCKDATA         43.   T  O  O  2  2 IF
 8.   F  1  O  5  4 BYTE              44.   F  O  O  5  8 IMPLICIT
 9.   T  O  O  2  4 CALL              45.   F  1  O 11  7 INCLUDE
10.   F  O  O  5  9 CHARACTER         46.   T  O  O  7  7 INQUIRE
11.   T  1  O  7  5 CLOSE             47.   F  O  O  5  7 INTEGER
12.   F  O  O  4  6 COMMON            48.   F  O  O  4  9 INTRINSIC
13.   F  O  O  5  7 COMPLEX           49.   F  O  O  5  7 LOGICAL
14.   F  O  O 12  8 CONTINUE          50.   F  O  1  9  8 NAMELIST
15.   F  O  O  6  4 DATA              51.   T  1  O  7  4 OPEN
16.   F  O  O 12  5 DEBUG             52.   F  1  O  4  9 PARAMETER
17.   T  1  O  7  6 DECODE            53.   T  O  O  2  5 PAUSE
18.   F  O  O  7 10 DEFINEFILE        54.   T  O  O  7  5 PRINT
19.   T  1  O  7  6 DELETE            55.   F  1  O  3  7 PROGRAM
20.   F  O  O  4  9 DIMENSION         56.   T  O  O  7  4 READ
21.   F  O  O 12  7 DISPLAY           57.   F  O  O  5  4 REAL
22.   F  1  O  5 13 DOUBLECOMPLEX     58.   T  O  O  2  6 RETURN
23.   F  O  O  5 15 DOUBLEPRECISION   59.   T  O  O  7  6 REWIND
24.   T  O  O 10  7 DOWHILE           60.   T  1  O  7  7 REWRITE
25.   T  O  O  2  2 DO                61.   F  O  O  4  4 SAVE
26.   F  O  O 11  5 EJECT             62.   T  O  O  2  4 STOP
27.   T  O  O 10  6 ELSEIF            63.   F  O  O  3 10 SUBROUTINE
28.   F  O  O 10  4 ELSE              64.   F  O  O 10  4 THEN
29.   T  1  O  7  6 ENCODE            65.   F  O  O 12  8 TRACEOFF
30.   F  O  O 10  5 ENDDO             66.   F  O  O 12  7 TRACEON
31.   F  O  O 12  8 ENDDEBUG          67.   T  O  O  7  4 TYPE
32.   T  O  O  7  7 ENDFILE           68.   T  O  O  7  4 WAIT
33.   F  O  O 10  5 ENDIF             69.   T  O  O  7  5 WRITE
34.   F  O  O  3  3 END               70.   F  O  O 13  9 UNDECODED
35.   F  O  O  3  5 ENTRY             71.   T  1  O  7  6 UNLOCK
36.   F  O  O  4 11 EQUIVALENCE       72.   F  1  O  4  7 VIRTUAL
```

Figure 3-1. Keywords File

Table 3-19. Statement Class Definitions

Statement Class	Statement Definition
1	Assignment
2	Control
3	Subprogram
4	Specification
5	Type Specification
6	DATA
7	Input/Output
8	FORMAT
9	NAMELIST
10	Structure
11	INCLUDE
12	IBM Debug
13	Undecoded, CONTINUE
14	Arithmetic Statement Function Definition

through 3-18. The statistical weights file contains a weight to be associated with the statistics having the indicated range of indexes. An example of a statistical weights file is the default file shown in Figure 3-2. The default statistical weights file location on the VAX-11/780 is DBB1:[TOOLS]WEIGHTS.SAP and is in the PANVALET library GJFEM.SAP.PAN on the IBM 4341.

```
DATE: 10:13:44- 3/27/85 WEIGHTS.SAP;1          CREATE:18-MAR-1985 PAGE:    1
                                                 USRDISK:<TOOLS.TOOO5.SAP>

 1.      1  256      0.0
 2.     12   12      1.0
 3.     50   50      0.5
 4.     58   58      4.0
 5.    122  122     10.0
 6.    173  173      2.0
 7.    233  233      2.0
 8.    246  246      2.0
```

Figure 3-2. Default Statistical Weights File

The user may specify another statistical weights file by using the /UW control switch (Section 2.4.1) on the VAX-11/780. If this switch is specified, SAP will prompt the user for the name of an alternative statistical weights file before processing the specified input file. The format for a statistical weights file is (2I5,F6.1). The remainder of each record may be used for comments. The first two columns specify the range of statistical weight indexes to be assigned the statistical weight (column 3). The weights are assigned to each specified range in the order in which the records appear in the file. For example, the first record in the default statistical weights file (Figure 3-2) sets all the weights (numbers 1 through 256) to an initial value of 0. The second record assigns a weight of 1.0 to the number of code lines in a module and so forth. The statistical weight indexes for particular statistics can be found in the

locator table pairs (Tables 3-1 through 3-18). The statistical weights file is also discussed in Appendix D.

3.5 HALSTEAD'S SOFTWARE SCIENCE METRICS

SAP calculates several of the quantities defined by Halstead (Reference 9). The quantities are all based on the following five measures:

- n_1 = number of unique or distinct operators appearing in the module
- n_2 = number of unique or distinct operands appearing in the module
- N_1 = total usage of all operators appearing in the module
- N_2 = total usage of all operands appearing in the module
- n_2^* = number of unique input/output parameters to the module

Appendix B presents the details of the methods used by SAP to collect these five measures.

The quantities calculated by SAP and the formulas used are as follows:

Program length	$N = N_1 + N_2$
Predicted program length	$\hat{N} = n_1 \times \log_2 n_1 + n_2 \times \log_2 n_2$
Program volume	$V = N \times \log_2 (n_1 + n_2)$
Potential program volume	$V^* = (2 + n_2^*) \times \log_2 (2 + n_2^*)$
Program level	$L = V^*/V$
Language level	$\lambda = V^{*2}/V$
Predicted effort	$E = V/L$
Predicted time	$\hat{T} = E/S$
Predicted bugs	$\hat{B} = V/E_0$

where S is the Stroud number (Reference 15) (64,800 mental discriminations per hour) and E_0 is the mean error rate (3,000 discriminations between potential programming errors).

The module summary page (Section 2.4.2) presents the counts of unique operators and operands, the total counts of operators and operands, the program and language level, and the predicted program length and effort required. The project summary (Section 2.4.1.3) also presents the count of input/output parameters, the program volume and the potential volume, and the predicted time and bugs. The module directory (Section 2.4.1.1) presents the total usage of operators and operands.

The /HL control switch (Section 2.1) can be used to present a detailed listing of the particular operators and operands detected by SAP.

3.6 McCABE'S COMPLEXITY MEASURE

SAP computes the cyclomatic complexity of a software module. According to McCabe (Reference 10), the cyclomatic complexity is calculated from the following formula:

$$\text{Cyclomatic Complexity: } V = d + 1$$

where d is the number of decisions in the module.

Appendix C presents the details of the methods used to count the number of decisions.

The cyclomatic complexity appears in the module directory (Section 2.4.1.1), the project summary (Section 2.4.1.3), and the module summary (Section 2.4.2).

APPENDIX A - FORTRAN STATEMENT ANALYSIS

A.1 DISCUSSION

This appendix presents information about how SAP processes each FORTRAN statement type. Tables A-1 through A-4 present an overview of this information.

Each table indicates for each statement type whether it is acceptable to standard FORTRAN (FORTRAN 77, Reference 1), DEC FORTRAN (References 2 through 5), or to IBM FORTRAN (References 6 and 7). Several Structured FORTRAN (SFORT) statements are also available on the DEC and IBM computers through the use of a preprocessor (Reference 8). The NAMELIST statement is also available on DEC computers through the use of the NAMELIST Preprocessor Program (NPP).

The tables also indicate how SAP and FORTRAN 77 classify each statement as executable or nonexecutable. This definition of "executable" for SAP is based on the contents of the keywords file as described in Section 3.3 and shown in Figure 3-1.

The last six columns of each table indicate how each statement is processed during the SAP analysis of Halstead operands and operators. An explanation of the terms used in the column headings in these tables is given in Appendix B.

The following section presents the individual statements recognized by SAP. The syntax for each statement is presented in the same format as it is presented in the FORTRAN standard (Reference 1), if the statement is acceptable to the standard. If the statement is not acceptable to all dialects, the dialect(s) to which it belongs is indicated (F77, DEC, IBM, SFORT, or NPP).

Unless specifically noted, SAP scans all delimiters and tokens in each statement.

Table A-1. Assignment/Control Statement Type Summary

STATEMENT TYPE	FORTRAN 77	DEC	IBM	EXECUTABLE-SAP	EXECUTABLE-F77	OPERAND ANALYSIS	OPERATOR ANALYSIS				KEYWORD (EOS)
							DELIMITER	KEYWORD	PROCEDURE	TRANSFER	
ASSIGNMENT, ARITHMETIC	●	●	●	●	●	●	●		●		●
ASSIGNMENT, LOGICAL	●	●	●	●	●	●	●		●		●
ASSIGNMENT, CHARACTER	●	●		●	●	●	●		●		●
ASSIGN	●	●	●	●	●			●			●
CALL	●	●	●	●	●	●	●		●	●	●
DOWHILE		●	a	●		●	●	●	●		●
DO	●	●	●	●	●	●		●	●		●
ELSEIF	●	●	a	●	●	●	●	●	●		●
GOTO, UNCONDITIONAL	●	●	●	●	●	●				●	●
GOTO, COMPUTED	●	●	●	●	●	●	●			●	●
GOTO, ASSIGNED	●	●	●	●	●	●				●	●
IF, ARITHMETIC	●	●	●	●	●	●	●	●	●	●	●
IF, BLOCK	●	●		●	●	●	●	●	●		●
IF, LOGICAL	●	●	●	●	●	●	●	●	●		●
IF, STRUCTURED		b	a	●		●	●	●	●		●
PAUSE	●	●	●	●	●			●			
RETURN	●	●	●	●	●		●	●	●		
STOP	●	●	●	●	●			●			
ELSE	●	●	a		●						
ENDDO		●	a								●
ENDIF	●	●	a		●						●
THEN											

[a]WITH IBM VERSION OF SFORT.

[b]WITH DEC VERSION OF SFORT.

Table A-2. Input/Output Statement Type Summary

STATEMENT TYPE	FORTRAN 77	DEC	IBM	EXECUTABLE-SAP	EXECUTABLE-F77	OPERAND ANALYSIS[a]	OPERATOR ANALYSIS				KEYWORD (EOS)[c]
							DELIMITER	KEYWORD	PROCEDURE	TRANSFER[b]	
ACCEPT		●		●		●	●	●	●	●	●
BACKSPACE	●	●	●	●	●	●	●	●	●	●	●
CLOSE	●	●	●	●	●	●	●	●	●	●	●
DECODE		●		●		●	●	●	●	●	●
DEFINEFILE		●	●			●	●	●	●	●	●
DELETE		●		●		●	●	●	●	●	●
ENCODE		●		●		●	●	●	●	●	●
ENDFILE	●	●	●	●	●	●	●	●	●	●	●
FIND		●	●	●		●	●	●	●	●	●
INQUIRE	●	●	●	●	●	●	●	●	●	●	●
OPEN	●	●	●	●	●	●	●	●	●	●	●
PRINT	●	●	●	●	●	●	●	●	●	●	●
READ	●	●	●	●	●	●	●	●	●	●	●
REWIND	●	●	●	●	●	●	●	●	●	●	●
REWRITE		●		●		●	●	●	●	●	●
TYPE		●		●		●	●	●	●	●	●
UNLOCK		●		●		●	●	●	●	●	●
WAIT			●	●			●	●	●	●	●
WRITE	●	●	●	●	●	●	●	●	●	●	●
FORMAT	●	●	●								
NAMELIST		d	●								

8721-191*dl/85

[a]ERR= OR END= STATEMENT LABELS ONLY.

[b]ERR= OR END= IF FOUND.

[c]COUNTED ONLY IF AN ERR= OR END= IS FOUND.

[d]WITH NAMELIST PREPROCESSOR PROGRAM.

Table A-3. Specification/Typing Statement Type Summary

STATEMENT TYPE	FORTRAN 77	DEC	IBM	EXECUTABLE-SAP	EXECUTABLE-F77	OPERAND ANALYSIS	OPERATOR ANALYSIS				KEYWORD (EOS)
							DELIMITER	KEYWORD	PROCEDURE	TRANSFER	
COMMON	●	●	●				●	●			
DATA	●	●	●				●	●			
DIMENSION	●	●	●				●				
EQUIVALENCE	●	●	●				●	●			
EXTERNAL	●	●	●								
INTRINSIC	●	●	●								
PARAMETER	●	●	●				●	●			
SAVE	●	●					●	●			
VIRTUAL		●									
BYTE		●					●				
CHARACTER	●	●	●				●				
COMPLEX	●	●	●				●				
DOUBLECOMPLEX		●					●				
DOUBLEPRECISION	●	●	●				●				
IMPLICIT	●	●	●								
INTEGER	●	●	●				●				
LOGICAL	●	●	●				●				
REAL	●	●	●				●				

8721-/91•dl/85

Table A-4. Subprogram and Other Statement Type Summary

STATEMENT TYPE	FORTRAN 77	DEC	IBM	EXECUTABLE-SAP	EXECUTABLE-F77	OPERAND ANALYSIS	OPERATOR ANALYSIS				KEYWORD (EOS)
							DELIMITER	KEYWORD .	PROCEDURE	TRANSFER	
ARITHMETIC STATEMENT FUNCTION DEFINITION	●	●	●								
BLOCKDATA	●	●	●								
CONTINUE	●	●	●		●						
END	●	●	●		●	●	●		●		●
ENTRY	●	●	●					●			
FUNCTION	●	●	●								
PROGRAM	●	●	●								
SUBROUTINE	●	●	●								
INCLUDE		●	●								
UNDECODED											

8721-(91*d)/85

A.2 FORTRAN STATEMENTS

A.2.1 ACCEPT STATEMENT

Syntax (DEC):

```
ACCEPT   f[,list]
```

or

```
ACCEPT   *[,list]
```

A.2.2 ARITHMETIC STATEMENT FUNCTION DEFINITION

Syntax:

```
fun ([d[,d]...]) = e
```

A.2.3 ASSIGN STATEMENT

Syntax:

```
ASSIGN   s   TO   i
```

A.2.4 ASSIGNMENT STATEMENT

Syntax:

```
v = e
```

No distinction is made between arithmetic, logical, or character assignment statements.

A.2.5 AT STATEMENT

Syntax (IBM):

```
AT stn
```

where stn is a statement label of an executable statement in the source. SAP counts AT as part of the prolog.

A.2.6 BACKSPACE STATEMENT

Syntax:

```
BACKSPACE   u
```

or

```
BACKSPACE   (alist)
```

where alist consists of selections from the following:

```
[UNIT=] u
IOSTAT = ios
ERR = s
```

A.2.7 BLOCKDATA STATEMENT

Syntax:

```
BLOCKDATA   [sub]
```

A.2.8 BYTE STATEMENT

Syntax (DEC):

```
BYTE v[/clist/][,v[/clist/]...
```

All names are marked as numeric.

A.2.9 CALL STATEMENT

Syntax:

```
CALL sub [([a[,a]...])]
```

A.2.10 CHARACTER STATEMENT

Syntax:

```
CHARACTER[*len[,]] nam[,nam]...
```

where nam is one of the following:

```
v[*len]
a[(d)][*len]
```

All names are marked as character.

A.2.11 CLOSE STATEMENT

Syntax:

```
CLOSE (cllist)
```

where cllist consists of selections from the following:

```
[UNIT=] u
IOSTAT = ios
```

```
    ERR = s
    STATUS = sta
```

A.2.12 COMMON STATEMENT

Syntax:

```
    COMMON [/[cb]/] nlist [[,]/[cb]/nlist]...
```

SAP tags each cb as a COMMON name and tags each name in each nlist as a COMMON variable name. All names are marked as array or nonarray.

A.2.13 COMPLEX STATEMENT

Syntax (F77):

```
    COMPLEX v[,v]...
```

Syntax (DEC, IBM):

```
    COMPLEX[*n] v[/clist/][[,]v[/clist/]...
```

All names are marked as numeric.

A.2.14 CONTINUE STATEMENT

Syntax:

```
    CONTINUE
```

SAP does not process beyond the CONTINUE keyword.

A.2.15 DATA STATEMENT

Syntax:

```
    DATA nlist /clist/[[,]nlist/clist/]...
```

A.2.16 DEBUG STATEMENT

Syntax (IBM):

```
    DEBUG option,...,option
```

SAP counts DEBUG as part of the prolog, where option consists of selections from the following:

UNIT(un)
SUBCHK(a_1,a_2,...,a_n)
TRACE
INIT(a_1,a_2,...,a_n)
SUBTRACE

A.2.17 DECODE STATEMENT

Syntax (DEC):

```
DECODE (c,f,b[,IOSTAT=ios][,ERR=s]) list
```

A.2.18 DEFINEFILE STATEMENT

Syntax (DEC):

```
DEFINEFILE u(m,n,U,v)[,u(m,n,U,v)]...
```

Syntax (IBM):

```
DEFINEFILE u(m,n,f,v)[,u(m,n,f,v)]...
```

A.2.19 DELETE STATEMENT

Syntax (DEC):

```
DELETE([UNIT=]u[,REC=r][,IOSTAT=ios][,ERR=s])
```

or

```
DELETE(u'r[,IOSTAT=ios][,ERR=s])
```

A.2.20 DIMENSION STATEMENT

Syntax:

```
DIMENSION a(d)[,a(d)]...
```

All names are flagged as arrays.

A.2.21 DISPLAY STATEMENT

Syntax (IBM):

```
DISPLAY list
```

where list is a list of variable names separated by commas. SAP counts DISPLAY as part of the prolog.

A.2.22 DO STATEMENT

Syntax (IBM, F77):

DO s[,]i = e_1, e_2[,e_3]

Syntax (DEC):

DO [s[,]]i = e_1, e_2[,e_3]

Processing of this statement includes loop nesting calculations.

A.2.23 DOUBLECOMPLEX STATEMENT

Syntax (DEC):

DOUBLECOMPLEX v[/clist/][,v[/clist/]...

All names are marked as numeric.

A.2.24 DOUBLEPRECISION STATEMENT

Syntax (F77):

DOUBLEPRECISION v[,v]...

Syntax (DEC, IBM):

DOUBLEPRECISION v[/clist/][,v[/clist]...

All names are marked as numeric.

A.2.25 DOWHILE STATEMENT

Syntax (DEC):

DO [s[,]] WHILE (e)

Syntax (SFORT):

DOWHILE(e)

Processing of this statement includes loop nesting calculations.

A.2.26 EJECT STATEMENT

Syntax (IBM):

```
    EJECT
```

A.2.27 ELSE STATEMENT

Syntax:

```
    ELSE
```

SAP does not process beyond the ELSE keyword.

A.2.28 ELSEIF STATEMENT

Syntax (DEC, IBM, F77):

```
    ELSEIF(e)THEN
```

A.2.29 ENCODE STATEMENT

Syntax (DEC):

```
    ENCODE(c,f,b[,IOSTAT=ios][,ERR=s]) list
```

A.2.30 END STATEMENT

Syntax:

```
    END
```

SAP does not process beyond the END keyword. Module statistic collection is complete when this statement is detected.

A.2.31 END DEBUG STATEMENT

Syntax (IBM):

```
    END DEBUG
```

SAP counts END DEBUG as part of the prolog.

A.2.32 ENDDO STATEMENT

Syntax (DEC, SFORT):

```
ENDDO
```

SAP does not process beyond the ENDDO keyword.

A.2.33 ENDFILE STATEMENT

Syntax:

```
ENDFILE u
```

or

```
ENDFILE (alist)
```

where alist consists of selections from the following:

```
[UNIT=] u
IOSTAT = ios
ERR = s
```

A.2.34 ENDIF STATEMENT

Syntax:

```
ENDIF
```

SAP does not process beyond the ENDIF keyword. SAP performs some calculations on block IF nesting.

A.2.35 ENTRY STATEMENT

Syntax:

```
ENTRY en [([d[,d]...])]
```

The name en is flagged as an ENTRY name. Each d is flagged as an argument to the module. An * appearing as one of the arguments is ignored.

A.2.36 EQUIVALENCE STATEMENT

Syntax:

```
EQUIVALENCE (nlist)[,(nlist)]...
```

All names appearing in nlist are marked as equivalenced.

A.2.37 EXTERNAL STATEMENT

Syntax:

```
EXTERNAL proc[,proc]...
```

All procs are marked as EXTERNAL.

A.2.38 FIND STATEMENT

Syntax (IBM):

```
FIND (u'r)
```

Syntax (DEC):

```
FIND (u'r[,IOSTAT=ios][,ERR=s])
```

or

```
FIND ([UNIT=]u,REC=r[,IOSTAT=ios][,ERR=s])
```

A.2.39 FORMAT STATEMENT

Syntax:

```
FORMAT fs
```

SAP does not process beyond the FORMAT keyword.

A.2.40 FUNCTION STATEMENT

Syntax:

```
[typ] FUNCTION fun[*n] ([d][,d]...)
```

Each d is flagged as an argument to the module. An * appearing as one of the arguments is ignored. The name fun is used as the module name.

A.2.41 GOTO STATEMENT

Syntax:

GOTO s	unconditional
GOTO (s[,s]...)[,]i	computed
GOTO i[[,](s[,s]...)]	assigned

The index of the computed GOTO is assumed to be a single unsubscripted variable name. The assigned GOTO is not examined beyond index i.

A.2.42 IF STATEMENT

Syntax (DEC, IBM, F77):

IF(e) s_1, s_2, s_3	arithmetic
IF(e) st	logical
IF(e) THEN	block

The statement labels of the arithmetic IF are not examined by SAP. The object statement of the logical IF, st, is parsed on a second pass. A logical IF is thus counted as two statements.

A.2.43 .IF STATEMENT

Syntax (SFORT):

.IF(e)	structured

A.2.44 IMPLICIT STATEMENT

Syntax:

IMPLICIT typ (a[,a]...)[,typ(a[,a]...)]...

Character ranges, a, for type CHARACTER are the only ranges noted by SAP. Any untyped variable name beginning with a character in a CHARACTER range is flagged as a CHARACTER variable name.

A.2.45 INCLUDE STATEMENT

Syntax (DEC):

```
INCLUDE 'filespec'[/[NO]LIST]
```

Syntax (IBM):

```
INCLUDE (name)
```

or

```
&&INCLUDE name
```

When the /XP switch is off, SAP does not process beyond the INCLUDE keyword. When the /XP switch is on, the source code contained in filespec is processed by SAP before proceeding to the following statements. SAP does not examine the /LIST or /NOLIST switch.

A.2.46 INQUIRE STATEMENT

Syntax:

```
INQUIRE (ilist)
```

where ilist consists of selections from the following:

```
[UNIT=]u
FILE = fin
IOSTAT = ios
ERR = s
EXIST = ex
OPENED = od
NUMBER = num
NAMED = nmd
NAME = fn
ACCESS = acc
SEQUENTIAL = seq
DIRECT = dir
FORM = fm
FORMATTED = fmt
```

```
UNFORMATTED = unf
RECL = rcl
NEXTREC = nr
BLANK = blnk
```

In the DEC dialect of FORTRAN, the following may also be used:

```
CARRIAGECONTROL = cc
KEYED = kyd
ORGANIZATION = org
RECORDTYPE = rty
```

A.2.47 INTEGER STATEMENT

Syntax (F77):

```
INTEGER v[,v]...
```

Syntax (DEC, IBM):

```
INTEGER[*n] v[/clist/][,v[/clist/]...
```

All names are marked as numeric.

A.2.48 INTRINSIC STATEMENT

Syntax:

```
INTRINSIC fun[,fun]...
```

SAP does not process beyond the INTRINSIC keyword.

A.2.49 LOGICAL STATEMENT

Syntax (F77):

```
LOGICAL v[,v]...
```

Syntax (DEC, IBM):

```
LOGICAL[*n] v[/clist][,v[/clist/]...
```

All names are marked as numeric.

A.2.50 NAMELIST STATEMENT

Syntax (IBM, NPP):

```
NAMELIST /nam/list[,/nam/list]...
```

SAP does not process beyond the NAMELIST keyword.

A.2.51 OPEN STATEMENT

Syntax:

```
OPEN (olist)
```

where olist consists of selections from the following:

```
[UNIT=] u
IOSTAT = ios
ERR = s
FILE = fin
STATUS = sta
ACCESS = acc
FORM = fm
RECL = rl
BLANK = blnk
```

In the DEC dialect of FORTRAN, the following may also be used:

```
ASSOCIATEVARIABLE = asv
BLOCKSIZE = blks
BUFFERCOUNT = bfr
CARRIAGECONTROL = cc
DISP = di
DISPOSE = dis
EXTENDSIZE = ext
INITIALSIZE = ini
KEY = key
MAXREC = mrc
NAME = nam
NOSPANBLOCKS = nos
```

```
    ORGANIZATION = org
    READONLY = rd
    RECORDSIZE = rsz
    RECORDTYPE = rty
    SHARED = shr
    TYPE = typ
    USEROPEN = uop
```

A.2.52 PARAMETER STATEMENT

Syntax:

```
    PARAMETER (p=e[,p=e]...)
```

Syntax (also for DEC):

```
    PARAMETER p=e[,p=e]...
```

A.2.53 PAUSE STATEMENT

Syntax:

```
    PAUSE [n]
```

Syntax (also, for IBM):

```
    PAUSE 'message'
```

A.2.54 PRINT STATEMENT

Syntax:

```
    PRINT f[,iolist]
```

Syntax (also DEC, F77):

```
    PRINT *[,iolist]
```

A.2.55 PROGRAM STATEMENT

Syntax:

```
    PROGRAM pgm
```

The name pgm is used as the module name. A main program module that does not contain a PROGRAM statement has a default name of .MAIN.

A.2.56 READ STATEMENT

Syntax:

```
READ (clist) [iolist]
```

or

```
READ f [,iolist]
```

Syntax (also DEC, IBM):

```
READ (u'r[,f][,ERR=s]) [iolist]
```

where clist consists of selections from the following:

```
[UNIT=] u
[FMT=] f
REC = rn
IOSTAT = ios
ERR = s1
END = s2
```

A.2.57 REAL STATEMENT

Syntax (F77):

```
REAL v[,v]...
```

Syntax (DEC, IBM):

```
REAL[*n] v[/clist/][,v[/clist]...
```

All names are marked as numeric.

A.2.58 RETURN STATEMENT

Syntax:

```
RETURN [e]
```

If the RETURN keyword is not the last item in the statement, a RETURN I is counted.

A.2.59 REWIND STATEMENT

Syntax:

```
REWIND u
```

or

```
REWIND [alist]
```

where alist consists of selections from the following:

```
[UNIT=] u
IOSTAT = ios
ERR = s
```

A.2.60 REWRITE STATEMENT

Syntax (DEC):

```
REWRITE (clist) [iolist]
```

where clist consists of selections from the following:

```
[UNIT=] u
[FMT=] f
IOSTAT = ios
ERR = s
```

A.2.61 SAVE STATEMENT

Syntax:

```
SAVE [a[,a]...]
```

A.2.62 STOP STATEMENT

Syntax:

```
STOP [n]
```

Syntax (also, for IBM):

```
STOP ['message']
```

SAP does not process beyond the STOP keyword.

A.2.63 SUBROUTINE STATEMENT

Syntax:

```
SUBROUTINE nam [([d[,d]...])]
```

Each d is flagged as an argument to the module. An * appearing as one of the arguments is ignored. The name nam is used as the name of the module.

A.2.64 TRACE OFF STATEMENT

Syntax (BIM):

```
TRACE OFF
```

SAP counts TRACE OFF as part of the prolog.

A.2.65 TRACE ON STATEMENT

Syntax (IBM):

```
TRACE ON
```

SAP counts TRACE ON as part of the prolog.

A.2.66 THEN STATEMENT

This statement is not a valid FORTRAN statement in any dialect acceptable to SAP. It is an artificial construct used at one time to prevent SAP from diagnosing a syntax error when parsing a block IF statement.

A.2.67 TYPE STATEMENT

Syntax (DEC):

```
TYPE f[,list]
```

or

```
TYPE *[,list]
```

A.2.68 UNLOCK STATEMENT

Syntax (DEC):

```
UNLOCK u
```

or

```
UNLOCK (alist)
```

where alist consists of selections from the following:

```
[UNIT=] u
IOSTAT = ios
ERR = s
```

A.2.69 VIRTUAL STATEMENT

Syntax (DEC):

```
VIRTUAL a(d)[,a(d)]...
```

All names are flagged as array.

A.2.70 WAIT STATEMENT

Syntax (IBM):

```
WAIT (clist) [iolist]
```

where clist consists of selections from the following:

```
[unit=] un
ID = i0
COND = i1
NUM = i2
```

A.2.71 WRITE STATEMENT

Syntax:

```
WRITE(clist) [iolist]
```

or

```
WRITE f [,iolist]
```

Syntax (also DEC, IBM):

```
WRITE(u'r[,f][,ERR=s]) [iolist]
```

where clist consists of selections from the following:

```
[UNIT=] u
[FMT=] f
REC = rn
IOSTAT = ios
ERR = s1
END = s2
```

APPENDIX B - HALSTEAD'S MEASURES IN SAP

B.1 INTRODUCTION

The measures and counts described by Halstead (Reference 9) are accumulated by SAP as each statement is analyzed. The basic quantities measured during analysis by SAP are as follows:

- n_1 = number of unique or distinct operators appearing
- n_2 = number of unique or distinct operands appearing
- N_1 = total usage of all operators appearing
- N_2 = total usage of all operands appearing
- n_2^* = number of unique input/output parameters to the module

The software metrics calculated from these five basic quantities are described in Section 3.5 and in Reference 9.

This appendix describes how each of the basic quantities is counted. This information is provided to assist researchers in judging how to interpret and apply the results reported by SAP. The interpretation of the basic quantities or of the metrics derived from them is not presented here, but is discussed in Reference 9. Halstead also indicates (Reference 16) how some FORTRAN language structures should be broken down into operators and operands. Halstead's techniques form the basis for the manner in which SAP parses statements to obtain the counts of operators and operands.

The basic quantities are reported in three places in SAP printed output: the complexity paragraph of the module statistic report (Section 2.4.2 and Figure 2-5), the module directory (Section 2.4.1.1 and Figure 2-3), and the global

summary (Section 2.4.1.2 and Figure 2-4). In addition, a complete listing of the individual operators and operands detected by SAP is available for each module through the use of the /HL switch (Section 2.4.2 and Figure 2-6).

All five basic quantities are written to the ALL.SAP sequential output file (Section 2.4.4).

The following subsections (B.2 through B.4) describe how SAP collects the counts for operators, operands, and input/output parameters, respectively. Most of the tables that appear in these sections contain descriptions of FORTRAN statement syntax. The names and symbols used in these descriptions are taken from the ANSI FORTRAN standards publication (Reference 1). The statements that are not part of the ANSI standard are described as they appear in the VAX FORTRAN manual (Reference 4). Section B.5 presents a sample of source code and a detailed accounting of how the Halstead counts are obtained.

B.2 COUNTING HALSTEAD OPERATORS

SAP counts Halstead operators in four groups: delimiter operators, keyword operators, procedure operators, and transfer operators. The following subsections discuss each type of operator.

B.2.1 DELIMITER OPERATORS

Decomposition of each statement by SAP results in a table of delimiters and tokens for the statement. As portions of particular statements are parsed by SAP, delimiters in the table are tested for membership in the list of delimiter operators (Table B-1). Each occurrence of a delimiter operator in the indicated portion of the particular statements shown in Table B-2 is counted.

At the completion of processing for a module, each delimiter operator with a nonzero count is included in the count of

Table B-1. Delimiter Operators

Delimiter Symbol	Definition in Context
//	Character string concatenation
**	Exponentiation
*	Multiplication or list directed input/output format identifier or alternate return specifier
/	Division
+	Addition or positive or floating-point exponent sign
-	Subtraction or negative or floating-point exponent sign
=	Replacement
(	Grouping
,	Separator
&	Alternate return specifier
.NE.	Comparison, not equal
.LT.	Comparison, less than
.LE.	Comparison, less than or equal
.EQ.	Comparison, equal
.GE.	Comparison, greater than or equal
.GT.	Comparison, greater than
.AND.	Logical conjunction
.OR.	Logical inclusive disjunction
.XOR.	Logical nonequivalence
.EQV.	Logical equivalence
.NOT.	Logical negation
.NEQV.	Logical nonequivalence

Table B-2. Statement Types Examined for Delimiter Operators (1 of 2)

Statement Type	Syntax	Portion of Statement Examined for Delimiter Operators
Assignment	v=e	Entire statement
Arithmetic Statement Function Definition	fun([d[,d]...])=e	Entire statement
CALL	CALL sub[([a[,a]...])]	Entire statement including open parenthesis preceding the argument list if present
COMMON	COMMON [/[cb]/]	Enclosing slash pair, entire list
DATA	DATA nlist/clist/	Entire statement, including repetition symbols and implied DO delimiters
DIMENSION	DIMENSION nlist	Entire statement
DOWHILE	DO[s[,]]WHILE(e)	Entire statement following and including the open parenthesis preceding the expression
EQUIVALENCE	EQUIVALENCE (nlist)	Entire statement
ELSE	ELSE	No delimiter ever present
THEN	THEN	No delimiter ever present
ELSEIF	ELSEIF(e)THEN	Expression including the enclosing parentheses
Arithmetic IF	IF(e)s_1,s_2,s_3	Expression including the enclosing parentheses; the statement label list is not examined

Table B-2. Statement Types Examined for Delimiter Operators (2 of 2)

Statement Type	Syntax	Portion of Statement Examined for Delimiter Operators
Structured IF	.IF(e)	Expression including the enclosing parentheses
Computed GOTO	GOTO(s[,s]...)[,]i	Entire statement following the open parenthesis preceding the statement label list
Logical IF	IF(e)st	Expression including the enclosing parentheses; st is not examined as part of the analysis of the logical IF (it is analyzed as a separate statement on a separate pass)
Block IF	IF(e)THEN	Expression including the enclosing parentheses
PARAMETER	PARAMETER(p=e)	Entire statement
RETURN	RETURN[e]	Expression
SAVE	SAVE[a[a[,a]...]	Entire statement
Any I/O	IOkeyword[(clist)] (iolist)	Entire statement
Type	type(*) nlist	Entire statement

distinct operators appearing (n_1), and the total count of each delimiter operator is added to the count of total usage of operators appearing (N_1).

B.2.2 KEYWORD OPERATORS

The keyword operators are associated with the FORTRAN language structures. The keyword operators are counted each time the particular statement(s) associated with a keyword is encountered. Table B-3 lists the keyword operators as they are labeled on the operator/operand report (Figures 2-6 and B-3), their operation, and the statement types that cause the keyword to be counted.

At the completion of processing for a module, each keyword operator with a nonzero count is included in the count of distinct operators appearing (n_1), and the total count of each keyword operator is added to the count of total usage of operators appearing (N_1).

B.2.3 PROCEDURE OPERATORS

References to external procedures are classified as procedure operators. Procedure operators are counted each time a reference to an external function is detected within a portion of a particular statement. Table B-4 lists the statements and indicates the portion of each statement examined for procedure operators.

At the completion of processing for a module, each procedure operator detected is included in the count of distinct operators appearing (n_1), and the total count of each procedure operator is added to the count of total usage of operators appearing (N_1).

B.2.4 TRANSFER OPERATORS

The transfer operators are associated with FORTRAN branches. SAP records each transfer operator as a list of the tokens making up the operator. Table B-5 lists the six

Table B-3. Keyword Operators (1 of 2)

Label on Operator/ Operand Report	Operation	Statement Types Causing 1 Count
ASSIGNTO	Transfer selection	ASSIGN
COMMON	Ordering	COMMON
DATA	Initialization	DATA
DO=,,	Loop definition	DO
DOWHILE	Loop definition	DOWHILE
Implied DO I/O or DATA	Initialization, control	I/O list or DATA
ENTRY	Decision	Begin execution

EOS	End of statement	Assignment
		Arithmetic statement function definition
		ASSIGN
		CALL
		DO
		DOWHILE
		END
		ENDDO
		ENDIF
		Unconditional GOTO
		Computed GOTO
		Assigned GOTO
		Arithmetic IF
		Logical IF
		Structured IF
		Block IF
		ELSEIF
		I/O statements

EQUIVALENCE	Ordering	EQUIVALENCE
IF()	Decision	Logical IF
IF(),,	Decision	Arithmetic IF
.IF()	Decision	Structured IF Block IF
ELSEIF	Decision	ELSEIF
ELSE	Alternative	ELSE

Table B-3. Keyword Operators (2 of 2)

Label on Operator/ Operand Report	Operation	Statement Types Causing 1 Count
PARAMETER	Initialization	PARAMETER
PAUSE	Decision	Stop execution
RETURN	Decision	Stop execution
SAVE	Initialization	SAVE
STOP	Decision	Stop execution
I/O key-word	Data transfer	ACCEPT BACKSPACE CLOSE DECODE DEFINEFILE DELETE ENCODE ENDFILE FIND INQUIRE OPEN PRINT READ REWIND REWRITE TYPE WAIT WRITE UNLOCK END= ERR= IOSTAT=

Table B-4. Statement Types Examined for Procedure Operators

Statement Type	Syntax	Portion of Statement Examined for Procedure Operators
Assignment	v=e	Entire statement
Arithmetic Statement Function Definition	fun([d[,d]...])	Entire statement
CALL	CALL sub[([a,...,a])]	sub is counted
DO	DO[s] i=e_1, e_2 [,e_3]	Argument list loop limits
DOWHILE	DO[s[,]]WHILE(e)	Expression
ELSEIF	ELSEIF(e)THEN	Expression
ELSE	ELSE	Procedures never present
Computed GOTO	GOTO(s[,s]...)[,]i	Index
THEN	THEN	Procedures never present
RETURN	RETURN[i]	Index
Arithmetic IF	IF(e)s_1,s_2,s_3	Expression
Structured IF	.IF(e)	Expression
Logical IF	IF(e)st	Expression only; st is not examined as part of the analysis of the logical IF (it is analyzed as a separate statement on a separate pass)
Block IF	IF(e)THEN	Expression
I/O	I/O keyword (nlist) (iolist)	Control list parameters and iolist

Table B-5. Transfer Operators

Statement Type	Syntax	Stored Tokens Used To Identify a Unique Transfer Operator
Alternate Return	CALL sub[([a[,a]...])]	List = sub, a, ...,a, where each argument (a) in the token list is an alternate return specifier label; this operator exists only if at least one argument is an alternate return
Any I/O statement	IO Keyword (...[,END=s]...)	List = s
Any I/O statement	IO Keyword (...[,ERR=s]...)	List = s
Unconditional GOTO	GOTO s	List = s
Computed GOTO	GOTO(s[,s]...)[,]i	List = s,...,s
Assigned GOTO	GOTOi[[,](s[,s]...)]	List = i, where the statement label list is not included in the token list
Arithmetic IF	IF(e)s_1,s_2,s_3	List = s_1, s_2, s_3

types of transfer operators and describes the list of tokens extracted from each statement type. The list of tokens associated with each discovered occurrence of a transfer operator is compared with the list of stored tokens for operators of the same type (unconditional GOTOs, assigned GOTOs, computed GOTOs, ERR=, END=, or alternate returns). The transfer operator is counted if it is a recurrence, or it is added to the list with a count of 1 if it has not been previously identified.

A transfer operator must be identical in type of transfer, length of token list, and order of tokens in the list to be counted as a recurrence. Thus, none of the following sample statements would be counted as the same transfer operator:

Statement	Stored Token List
GOTO 100	100
GOTO 200	200
CALL ABC (*100,X,Y,*200)	ABC,100,200
CALL ABC (*200,X,Y,*100)	ABC,200,100
GOTO (100,200),I	100,200,I
GOTO (100,200,300),I	100,200,300,I
READ (5,100,ERR=200)X	200

At the completion of processing for a module, each transfer operator detected is included in the count of distinct operators appearing (n_1), and the count of each transfer operator is added to the count of total usage of operators appearing (N_1).

B.3 COUNTING HALSTEAD OPERANDS

The result of the decomposition of each statement by SAP is a table of delimiters and tokens. As portions of particular statements are parsed by SAP, each token in the table is flagged and counted as an operand. Each occurrence of a token that is not a function reference (procedure operator)

or a FORTRAN statement keyword is counted when it appears in the indicated portion of the particular statements shown in Table B-6. Because of the way in which SAP parses statements, character and Hollerith constants appearing in the indicated statements are not counted as operands.

At the completion of processing for a module, each flagged token is included in the count of distinct operands appearing (n_2), and the total count of each operand's use is added to the count of total usage of operands appearing (N_2).

B.4 COUNTING HALSTEAD INPUT/OUTPUT PARAMETERS

At the completion of module processing, SAP computes the number of input/output parameters to the module from the sum of the following three counts:

- Total number of variable names passed to the module in the SUBROUTINE or FUNCTION statement
- Total number of variable names appearing in ENTRY statements
- Number of variables and arrays appearing in COMMON statements that are also used in the module

These counts appear individually on the module statistics report.

B.5 COUNTING HALSTEAD OPERATORS AND OPERANDS: AN EXAMPLE

This section presents a sample module and an accounting for the resulting Halstead counts. This example should be used in conjunction with the procedures and tables presented in Sections B.2 and B.3 to understand how SAP performs this analysis.

Table B-6. Statement Types Examined for Operands (1 of 2)

Statement Type	Syntax	Portion of Statement Examined for Operands
ASSIGN	ASSIGN s TO i	s and i
Assignment	v=e	Entire statement
Arithmetic Statement Function Definition	fun([d[,d]...])=e	Entire statement
CALL	CALL sub[([a[,a]...])]	Argument list; not alternate return specifiers
COMMON	COMMON nlist	All variable names and array dimensions in nlist
DATA	DATA nlist/clist/	Variable and array names, constants, repetition factors, and implied DO
DIMENSION	DIMENSION nlist	All array names and dimension bounds in nlist
DO	DO[s]i=e_1,e_2[,e_3] or DO[S[,]]i=e_1,e_2[,e_3]	Entire statements to the right of, and including, i; s is not counted when present
DOWHILE	DO[s[,]]WHILE(e)	Expression; s is not counted when present
ELSE	ELSE	Operands never present
THEN	THEN	Operands never present
ELSEIF	ELSEIF(e)THEN	Expression
Arithmetic IF	IF(e)s_1,s_2,s_3	Expression; s's are not counted
Structured IF	.IF(e)	Expression
Logical IF	IF(e)st	Expression only; st is not examined as part of the analysis of the logical IF (it is analyzed as a separate statement on a separate pass)

Table B-6. Statement Types Examined for Operands (2 of 2)

Statement Type	Syntax	Portion of Statement Examined for Operands
Block IF	IF(e)THEN	Expression
EQUIVALENCE	EQUIV. (nlist)	All variable names and array indexes in nlist
Assigned GOTO	GOTOi[[,](s[,s]...)	i is counted; the s's are not counted
Computed GOTO	GOTO(s[,s]...)[,]i	Entire statement to the right of, not including, the statement labels
Unconditional GOTO	GOTOs	s is counted
PARAMETER	PARAMETER (p=e)	Parameter names and constants used in the expression
PAUSE	PAUSE(n)	n
RETURN	RETURN(e)	Expression
SAVE	SAVE (a)	Variable, array, and COMMON block names
STOP	STOP(n)	n
Any I/O	IO keyword ([clist]) [iolist]	Entire statement, excluding END=n, or ERR=n
Type	type[*n] nlist	All variable names and array dimensions in nlist

Figure B-1 contains the sample source code. Subroutine TDIST is used throughout this manual whenever a sample report describes an individual module. TDIST is also the last routine in all summary reports and file listings presented. Thus, this figure may also be used to understand other counts produced by SAP.

Figure B-2 shows a line-by-line summary of Halstead operator and operand counts. Column 1 of Figure B-2 indicates a line number from Figure B-1. Columns 2 through 5 indicate the specific operators identified on each line. These columns also contain line totals for each operator type. Column 6 shows the line totals for operators. Columns 7 and 8 present the specific operands and operand line totals, respectively. The last two lines of Figure B-2 show column totals for both unique occurrences and total usage. The totals for columns 6 and 8 correspond to n_1, N_1, n_2, and N_2.

Although Figure B-2 is not a report produced by SAP, Figure B-3 is the SAP report that summarizes the details of Halstead counting. This report is produced for each module when the /HL control switch is set on.

```
DATE: 10:13:50- 3/27/85 TDIST.FOR;1              CREATE:07-SEP-1984 PAGE:    1
                                                   USRDISK:<TOOLS.T0005.SAP>

 1.         SUBROUTINE TDIST (N, X, Y, DIST)
 2. C                                                    PASSED
 3.         INTEGER    N
 4.         REAL       X(N), Y(N), DIST
 5. C                                                    LOCAL
 6.         INTEGER    I, MSGNUM, K
 7.         REAL       XL, YL, DX, DY, X2, Y2, R2, R
 8.         LOGICAL    ERR
 9. C                                                    GLOBAL
10.         REAL       SQRT
11. C                                                    INITIALIZE
12.         XL = 0.0
13.         YL = 0.0
14.         DIST = 0.0
15. C                                                    FOR ALL POINTS
16.         DO 200 I=1, N
17.           DX = X(I) - XL
18.           X2 = DX*DX
19.           DY = Y(I) - YL
20.           Y2 = DY*DY
21. C                                                    CALC./CHECK SEPARATION
22.           R2 = X2 + Y2
23.           CALL VERIFY (R2, ERR)
24. C                                                    OBTAIN SEPARATION
25.           IF (ERR) THEN
26.             K = I - 1
27.             WRITE (6, 100, ERR=300) K, I
28.    100      FORMAT (1X, 'ERROR, POINTS ', I3, ' AND ', I3,
29.        *             ' TOO CLOSE')
30.             R = 0.0
31.           ELSE
32.             R = SQRT (R2)
33.           END IF
34. C                                                    ACCUMULATE
35.           DIST = DIST + R
36.           XL = X(I)
37.           YL = Y(I)
38.    200  CONTINUE
39. C                                                    NORMAL RETURN
40.         RETURN
41. C                                                    ERROR WRITING MESSAGE
42.    300  CONTINUE
43.         MSGNUM = 27
44.         CALL ERRMSG (MSGNUM, *400)
45.         RETURN
46. C                                                    UNABLE TO WRITE ANY
47. C                                                    MESSAGES, ABORT RUN
48.    400  CONTINUE
49.         CALL ABORT
50. C
51.         END
```

Figure B-1. Sample Source Code

	OPERATORS							OPERANDS	
LINE NO.	DELIMITER	CNT	KEYWORD	CNT	PROCEDURES	CNT	TRANSFER	CNT	TOTAL		TOTAL
100											
200											
300											
400											
500											
600											
700											
800											
900											
1000											
1100											
1200	=	1	EOS	1					2	XL O.O	2
1300	=	1	EOS	1					2	YL O.O	2
1400	=	1	EOS	1					2	DIST O.O	2
1500											
1600			DO=,, EOS	2					2	I 1 N	3
1700	= (-	3	EOS	1					4	DX X I XL	4
1800	= *	2	EOS	1					3	X2 DX DX	3
1900	= (-	3	EOS	1					4	DY Y I YL	4
2000	= *	2	EOS	1					3	Y2 DY DY	3
2100											
2200	= +	2	EOS	1					3	R2 X2 Y2	3
2300	,	1	EOS	1	VERIFY	1			3	R2 ERR	2
2400											
2500			.IF() EOS	2					2	ERR	1
2600	= -	2	EOS	1					3	K I 1	3
2700			EOS	1			ERR=300	1	2	300	1
2800											
2900											
3000	=	1	EOS	1					2	R O.O	2
3100			ELSE	1					1		
3200	= (	2	EOS	1	SQRT	1			4	R R2	2
3300											
3400											
3500	= +	2	EOS	1					3	DIST DIST R	3
3600	= (	2	EOS	1					3	XL X I	3
3700	= (	2	EOS	1					3	YL Y I	3
3800											
3900											
4000											
4100											
4200											
4300	=	1	EOS	1					2	MSGNUM 27	2
4400	, *	2	EOS	1	ERRMSG	1	ERRMSG 400	1	5	MSGNUM 400	2
4500											
4600											
4700											
4800											
4900					ABORT	1			1		
5000											
5100											
(UNIQUE)		(6)		(4)		(4)		(2)	(16)		(21)
TOTAL		30		23		4		2	59		50

Figure B-2. Line-by-Line Summary of Halstead Counts

SAMPLE.FOR/HL/SL/XP

HALSTEAD OPERATORS

DELIMITERS

0	//	0	**	3	*	0	/	2	+	3	-
15	=	9	(	15	,	0	&	0	.NE.	0	.LT.
0	.LE.	0	.EQ.	0	.GE.	0	.GT.	0	.AND.	0	.OR.
0	.XOR.	0	.EQV.	0	.NOT.	0	.NEQV.				

KEYWORDS

0	ASSIGNTO	0	COMMON	0	DATA	1	DO=,,	0	DOWHILE	0	IMPLD DO
1	ELSE	0	ELSE IF	0	ENTRY	23	EOS	0	EQUIVLNC	0	IF()
0	IF(),,	1	.IF()	0	PARAMETR	0	PAUSE	2	RETURN	0	SAVE
0	STOP	0	ACCEPT	0	BACKSPCE	0	CLOSE	0	DECODE	0	DEFNFILE
0	DELETE	0	ENCODE	0	ENDFILE	0	FIND	0	INQUIRE	0	OPEN
0	PRINT	0	READ	0	REWIND	0	REWRITE	0	TYPE	0	WAIT
1	WRITE	0	UNLOCK	1	ERR =	0	END =	0	IOSTAT =		

PROCEDURES: EXTERNAL FUNCTIONS AND SUBROUTINES

1	VERIFY	1	SQRT	1	ERRMSG	1	ABORT

TRANSFERS

1	ERR=	300
1	ALT.RET.	ERRMSG 400

HALSTEAD OPERANDS

2	1	8	I	3	K	2	N	4	R	3	X
3	Y	1	27	4	R2	3	X2	3	Y2	4	0.0
4	DX	4	DY	4	XL	4	YL	3	ERR	4	DIST
1	SQRT	3	MSGNUM								

Figure B-3. SAP Operand/Operator Summary Report

APPENDIX C - McCABE'S MEASURE IN SAP

C.1 CYCLOMATIC COMPLEXITY

McCabe's measure (also referred to as the cyclomatic complexity) is described in Reference 10 where it is proposed as an indicator of computer program complexity. The quantity measured is the number of linearly independent paths in a program. The measure is developed from graph theory in which the cyclomatic number V(G) is the maximum number of linearly independent circuits in a strongly connected graph. The expression for the cyclomatic number is

$$V(G) = e - n + 2p \qquad \text{(C-1)}$$

where e = number of edges
n = number of vertices
p = number of connected components (usually one)

McCabe shows that for a program with unique entry and exit nodes, the cyclomatic number for the program control graph is

$$V(G) = d + 1 \qquad \text{(C-2)}$$

where d = number of decisions in the program

The information in this appendix provides a basis that may be used by researchers to judge how and when to use the value of the cyclomatic complexity calculated by SAP. The interpretation of the cyclomatic number as a measure of program complexity is not presented here, but is discussed in Reference 10. McCabe also indicates how some of the FORTRAN language structures contribute to the cyclomatic complexity. McCabe's techniques form the basis for the manner in which SAP calculates the cyclomatic complexity as presented in Section C.3.

C.2 SAP REPORTS OF THE CYCLOMATIC COMPLEXITY

The cyclomatic complexity is reported in three places in SAP printed output: the complexity paragraph of the module statistics report (Section 2.4.2 and Figure 2-5), the module directory (Section 2.4.1.1 and Figure 2-3), and the global summary report (Section 2.4.1.2 and Figure 2-4).

The cyclomatic complexity is not written to the SAP sequential data files. However, a related number, the count of decisions, is written to the sequential output file ALL.SAP (Section 2.4.4). The cyclomatic complexity can be computed from the number of decisions by using Equation (C-2).

C.3 CALCULATION OF THE CYCLOMATIC COMPLEXITY

SAP calculates the cyclomatic complexity from Equation (C-2). The count of decisions in a source code module is obtained by SAP in two stages: (1) during source code parsing, individual contributions to the decision count by various statement types and constructs are accumulated, and (2) the contributions are combined after module parsing is completed. The number of decisions is calculated as

$$d = d(cg) + d(if) + d(do) + d(op) + d(i/o) + d(cl) \quad (C-3)$$

where
- d = number of decisions in the module
- $d(cg)$ = contribution from computed GOTO statements
- $d(if)$ = contribution from IF statements
- $d(do)$ = contribution from looping statements
- $d(op)$ = adjustment due to compound decisions (Section C.3.4)
- $d(i/o)$ = END= or ERR= in I/O statements
- $d(cl)$ = CALL with alternate returns

Each contribution is discussed below.

C.3.1 DECISION COUNT FROM COMPUTED GOTO STATEMENTS

The computed GOTO statement is treated as a CASE structure. The number of decisions in an individual computed GOTO statement is calculated from the number of statement labels in the statement label list. It is assumed that the computed GOTO index expression points to one of the statement labels. When this is so, the count of decisions is one less than the number of statement labels in the statement label list. The following computed GOTO would be counted as three decisions:

```
GOTO (100, 200, 300, 400), I
```

SAP obtains the count of decisions by starting after the open parenthesis and counting tokens (including the index expression). The count is then decreased by 2 (not 1) because the index expression is assumed to be a single unsubscripted variable name.

C.3.2 DECISION COUNT FROM IF STATEMENTS

The contribution to the decision count from IF statements is calculated by summing the occurrences of individual logical IF statements, structured IF statements, and block IF statements. Twice the count of individual arithmetic IF statements is added to the above sum.

C.3.3 DECISION COUNT FROM LOOPING STATEMENTS

The contribution to the decision count from looping statements is calculated by summing the occurrences of individual DO and DOWHILE statements.

C.3.4 DECISION COUNT ADJUSTMENT

An adjustment is made to the decision count because of compound decisions. As shown in Figure C-1, a single block IF statement with a compound expression (logical expressions using the .AND., .OR., .XOR., .EQV., or .NEQV. operators)

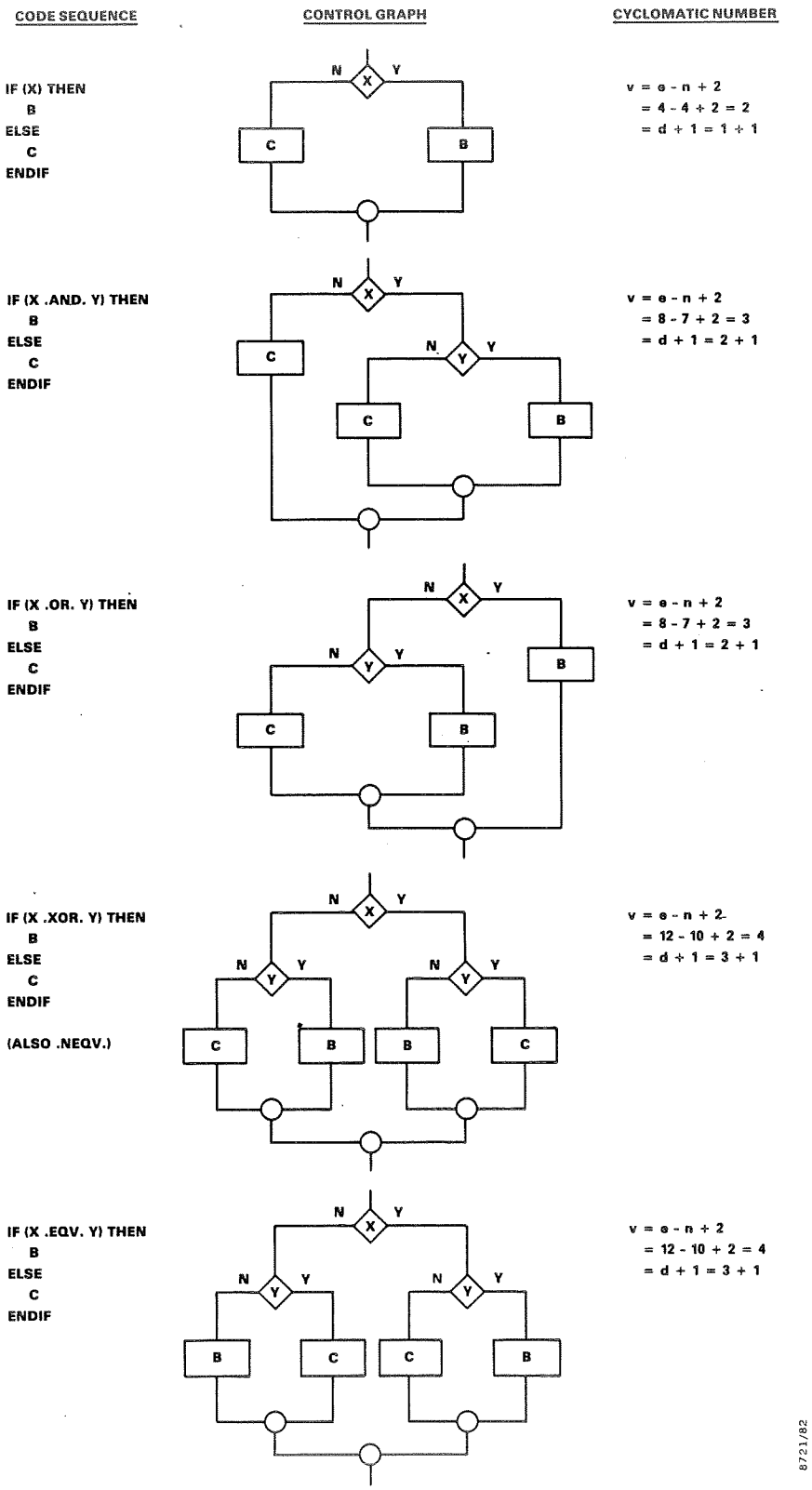


Figure C-1. Control Graphs and Cyclomatic Numbers for Compound Decisions

can be expressed as two or more block IF statements with simple logical expressions. A compound expression using an .AND. or .OR. has one "hidden" decision that is not counted by counting the occurrences of block IF statements. A compound expression using an .XOR., .EQV., or .NEQV. has two hidden decisions.

The counts of logical operator usage are obtained from examination of assignment statements and CALL, DOWHILE, computed GOTO, ENDIF, ELSEIF() THEN, ELSE, THEN, arithmetic IF, logical IF, structured IF, block IF, and ELSEIF statements. The use counts of .AND., .OR., and .XOR. are summed and used as the adjustment, unless they are used as masking operators or in combining logical values.

APPENDIX D - USER COMPLEXITY TECHNIQUES

D.1 INTRODUCTION

The SAP user has several options when considering how to compute individualized complexity measures based on the statistics gathered during module processing by SAP. Two options are discussed in this appendix: the SAP statistical weights file and user "stubs" UCPLX1 and UCPLX2. Other options, such as the design of programs to access the SAP sequential or data base files or even the direct modification of code within SAP, are available to the user but their descriptions are beyond the scope of this document.

D.2 USER'S STATISTICAL WEIGHTS FILE

The use of the statistical weights file is briefly discussed in Section 3.4. This file is used to compute a complexity composed of a selected set of SAP statistics in a linear combination. The weighted complexity is displayed in both the module directory (Section 2.4.1.1, Figure 2-3) and the module statistics pages (Section 2.4.2, Figure 2-5). The following example is used to demonstrate how a statistical weights file is designed.

In the example, the user wishes to compute, as a source code measure, the difference between the number of paths into and out of a module. The statistical weights file shown in Figure D-1 can be used to compute such a measure. This figure is referred to in the following explanation.

The first record in the file clears all of the statistical weights to 0. The format for this and all records in the file is 2I5, F6.1.

```
  1  256    0.0  Initialization
154  154    1.0  Count of END statements
155  155    1.0  Count of ENTRY statements
127  127   -1.0  Count of BLOCK DATA statements
178  178   -1.0  Count of RETURN statements
182  182   -1.0  Count of STOP statements
```

Figure D-1. Sample User Statistical Weights File

One way to calculate the sample measure is to use the counts of statement types described in Tables 3-7 and 3-8. The number of paths into the module might be calculated by summing the counts of SUBROUTINE, FUNCTION, and PROGRAM statements (statistical weight indexes 183, 160, and 175, respectively from column 8 of Table 3-8) and then adding the count of ENTRY statements (statistical weight index 155). However, this would leave uncounted the implied entrance at the start of a main routine having no PROGRAM statement. Instead, because each module type, except BLOCKDATA, always has at least one path into the module in addition to its ENTRY statements, the design shown in Figure D-1 uses the fact that each module also has exactly one END statement. Thus, the number of paths into the module is calculated from the number of END and ENTRY statements (statistical weight indexes 154 and 155) minus the number of BLOCKDATA statements (statistical weight index 127). Record numbers 2 through 4 in Figure D-1 thus account for the number of paths into a module.

The number of paths out of a module is the sum of the counts of RETURN and STOP statements (statistical weight indexes 178 and 182). These counts are subtracted from the number of paths into the module according to record numbers 5 and 6 of Figure D-1.

The user statistical weights file may be used whenever the source code measure can be expressed as a weighted sum of the statistics shown in Tables 3-1 through 3-14.

D.3 USER COMPLEXITY STUBS

The user may wish to compute a complexity measure that is not a weighted sum of selected statistics as described in Section D.2. This option is provided to the user through the use of two modules (stubs) currently called by SAP immediately after completing processing of a module. The two modules, UCPLX1 and UCPLX2, may be replaced with user-written routines. The supplied versions of these routines do not perform any calculations.

The user may have access to all COMMON block variables in SAP through the use of the INCLUDE statement. These variables and INCLUDE files are described in Reference 13. The results of any user calculation in these routines are passed to SAP through one argument to each routine. These arguments are output only and are of type REAL*4. The arguments are printed in the module directory (Section 2.4.1.1, Figure 2-3) with format F5.1.

Reference 13 contains information on how the SAP task image is linked. When the user has written either UCPLX1 or UCPLX2 and compiled the module, the object module should replace the supplied module and the SAP task image should be relinked.

To reduce the chance of introducing error into other SAP processes, the user should take care not to alter any variable in COMMON or to perform any input or output to SAP files (Reference 13).

APPENDIX E - QUALITY INDEX

The quality index in the complexity paragraph of the module statistics report (Section 2.4.2 and Figure 2-5) and the global statistics summary (Section 2.4.1.2 and Figure 2-4) predicts software quality. The SEL defines software quality as a function of productivity, error rate, and maintainability. None of this information is available to SAP, so another model based on the most likely programming style to reflect these parameters is used to compute software quality. This model is based on software quality research done in the SEL.

The quality index is defined as the sum

$$QI = ARrt + COrt + CMrt + DCrt + TY - VRrt$$

which is then normalized so that its range is from 0 to 100.

ARrt is defined as the ratio of arguments to source lines of code. Arguments is defined as the sum of the arguments from the SUBROUTINE, ENTRY, and CALL statements plus two times the number of variables in COMMON statements. This ratio is normalized so that it falls between 0.25 and 3 on a normal distribution curve. The optimum value is approximately 1 argument per 10 lines of source lines of code.

COrt is defined as the ratio of COMMON statements to source lines of code. This ratio is normalized so that it falls between 0.25 and 3 on a normal distribution curve. The optimum value is approximately one COMMON statement per 50 lines of source lines of code.

CMrt is defined as the ratio of comment lines (including in-line comments) to the total lines of code. This ratio is normalized so that it falls between 0.25 and 3 on a normal distribution curve. The optimum value is approximately

equal to comment lines comprising 45 percent of the total lines of code.

DCrt is defined as the ratio of decisions to source lines of code. Decisions is defined as the number of decisions as defined in Section C.3, plus the total number of subroutine and function calls. This ratio is normalized so that it falls between 0.25 and 3 on a normal distribution curve. The optimum value is approximately one decision per eight lines of source lines of code.

TY is defined as the number of different code types present in the source lines of code. The three code types are input/output, control, and assignment. Twenty percent or more of any code type in a module is significant enough to add one to the count of code types. The minimum value of TY is 1; the maximum value is 3.

VRrt is the ratio of the number of different types of variables (BYTEs, CHARACTER, COMPLEX, DOUBLECOMPLEX, DOUBLEPRECISION, INTEGER, LOGICAL, and REAL) plus the number of COMMON statements to source lines of code. This ratio is normalized so that it falls between 1 and 3 on a normal distribution curve. The optimum value is approximately 1 different type of variable (or less) per every 10 lines of source lines of code.

Before normalization from 0 to 100, the quality index would have a range of 0 to 12. This is easily seen by adding the maximum values of ARrt, COrt, CMrt, DCrt, TY, and VRrt, which is 15 - 3. Each parameter of the quality index had been normalized so that its upper bound was known, to normalize the quality index.

This quality index and software quality prediction model should not be taken as the definitive truth. Much research is still to be done in software quality measurement and prediction. Certainly this research will help to refine this

model. The quality index is included in this version of SAP to help aid in software quality research by producing an experimental measure that can be compared to other measures of software quality.

Reference 13 contains more information on how the quality index is computed.

APPENDIX F - SAP ERROR MESSAGES

SAP informs the user of abnormal conditions during execution by writing messages to the user's listing file. The errors detected by SAP are, in general, those encountered while opening or reading an external file, syntax errors in the source code encountered while parsing a statement, and conditions that cause an internal table to be exceeded.

Most SAP error messages appear in the following format:

*****routine type***** - msg

where routine = name of the SAP routine that detects the abnormal condition
type = ERROR or WARNING
msg = explanatory text

In general, a message of type ERROR indicates a condition that either causes SAP to stop processing for the module or the entire file, or causes the reported statistics to be misleading. A message of type WARNING indicates a condition that affects only the parsing of a single statement.

Each message originating from SAP is presented below. The message is presented and is followed by an explanation of the probable cause of the error. The messages are arranged according to the alphabetical order of the originating routine.

Message:
***** ADDPOT ERROR ***** - NOT ENOUGH ROOM TO ADD ITEM TO NODE, MAX=nnnn
Explanation: The transfer list table is not large enough to contain all transfer operators in the module.
Originating Subroutine: ADDPOT

Message:

--INVALID SWITCH--aaa

Explanation: The control switch (aaa) specified by the user could not be identified.

Originating Subroutine: CINPUT

Message:

*****-INVALID FILE-*****a...a

Explanation: The file name (a...a), supplied by the user in response to the SAP> prompt, could not be opened.

Originating Subroutine: CINPUT

Message:

*****ERROR OPENING DATA BASE FILE*****

Explanation: An attempt to open the file ALL.SAP resulted in an error.

Originating Subroutine: DEFSEL

Message:

GARBAGE COLLECTION BEING ATTEMPTED!!

Explanation: The symbol table is not large enough to contain all of the symbols in a module without first removing the deleted symbols.

Originating Subroutine: GARCOL

Message:

*****GLINE ERROR***** - READ ERROR ON LUN nn

Explanation: An error occurred while reading the source code input file.

Originating Subroutine: GLINE

Message:

*****GLINE ERROR***** - LINE LENGTH nnnnn.GT.100

Explanation: A source input line was found that exceeds the maximum allowed length.

Originating Subroutine: GLINE

Message:

*****HOPTR3 WARNING***** - NO ROOM LEFT FOR SUBR/ENTRY/FUNC aaaaaaaa. IGNORED

Explanation: The procedure operator table is not large enough to contain all procedure operators in the module.

Originating Subroutine: HOPTR3

Message:

*****HSCAN WARNING***** - HOLLERITH FIELD LONGER THAN LINE

Explanation: SAP cannot process a Hollerith field that is continued onto a continuation card.

Originating Subroutine: HSCAN

Message:

INCLUD TERMINATED DUE TO TOO MANY LEVELS

Explanation: SAP cannot expand INCLUDE cards beyond a depth of four.

Originating Subroutine: INCLUD

Message:

*****INVALID FILE***** a...a

The file name (a...a), supplied by the user in response to the SAP> prompt, could not be opened.

Originating Subroutine: INCLUD

Message:

OPEN ERROR - a...a

Explanation: An indirect file (a...a), supplied by the user in response to the SAP> prompt, could not be opened.

Originating Subroutine: INPUT

Message:

FILE READ ERROR

Explanation: An error occurred while reading a command line from an indirect file.

Originating Subroutine: INPUT

Message:

*****INTGR4 WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: A token that is used in a context in which a statement label is expected could not be converted to a binary integer value.

Originating Subroutine: INTGR4

Message:

*****LOADK ERROR***** - OPEN ERROR ON KEYWORD FILE

Explanation: An error occurred while opening the keyword file.

Originating Subroutine: LOADK

Message:

*****LOADK ERROR***** - READ ERROR ON KEYWORD FILE

Explanation: An error occurred while reading the keyword file.

Originating Subroutine: LOADK

Message:

*****LOADK ERROR***** - ERROR ON WEIGHTS FILE

Explanation: An error occurred while opening or reading either the default or user-specified weights file.

Originating Subroutine: LOADK

Message:

*****LOOKP ERROR-ILLEGAL SYMBOL TABLE ADDRESS*****

Explanation: The symbol table pointer passed to LOOKP does not point to the portion of the table in use.

Originating Subroutine: LOOKP

Message:

*****NEWPOT ERROR***** - NOT ENOUGH ROOM FOR A NEW NODE IN THE TRANSFER LIST, MAX=nnnn

Explanation: The transfer table is not large enough to contain all transfer operators in the module.

Originating Subroutine: NEWPOT

Message:

*****POKEP ERROR - ADDRESS OR LENGTH BAD IN SYMBOL TABLE STOW OPERATION

Explanation: The symbol table pointer passed to POKEP does not point within the bounds of the symbol table, or the symbol will not fit into the symbol table at the indicated position.

Originating Subroutine: POKEP

Message:

*****POKES ERROR***** - SYMBOL TABLE FULL

Explanation: The symbol table is not large enough to contain all the symbols identified in the module.

Originating Subroutine: POKES

Message:

*****PRCALL WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: Three conditions can cause this error: the CALL keyword is the only item in the statement, the subroutine name was used previously as other than a subroutine name, or the first item (if any) following the subroutine name is not an open parenthesis.

Originating Subroutine: PRCALL

Message:

PRDOS: STACK POINTER ERROR

Explanation: The stack used to store DO loop target labels is not large enough to store all the target labels in the module.

Originating Subroutine: PRDOS

Message:

PRDOS: ERRONEOUS DOWHILE STATEMENT

Explanation: Keyword WHILE in a DOWHILE statement is not followed by an open parenthesis, or DO loop control variable WHILE is not followed by an equal sign.

Originating Subroutine: PRDOS

Message:

*****PRGOTO WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: A constant follows the GOTO keyword and is not the last item in the statement, or the delimiter following the GOTO keyword is not an open parenthesis.

Originating Subroutine: PRGOTO

Message:

PRIFS: STRUCTURED .IF ERROR

Explanation: The closing parenthesis that follows the logical expression is not the last item in the statement.

Originating Subroutine: PRIFS

Message:

PRIFS: ERROR ON ELSE-IF-THEN

Explanation: The last item in the ELSEIF statement is not keyword THEN.

Originating Subroutine: PRIFS

Message:

SYNTAX ERROR IN IMPLICIT STATEMENT

Explanation: Four conditions can cause this error message: (1) the token following an asterisk indicating the length of a variable type cannot be converted to a binary integer; (2) the variable type is not recognized; (3) a delimiter is found in the alphabetic range specifier that is not an open parenthesis, a comma, a minus sign, or a close parenthesis; or (4) a delimiter is found where a variable type is expected.

Originating Subroutine: PRIMPL

Message:

*****PRIO WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: The open parenthesis immediately following the I/O keyword is not matched with a closing parenthesis.

Originating Subroutine: PRIO

Message:

*****PRSPEC WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: Three conditions may cause this error message: (1) a COMMON block name is not enclosed by slashes, (2) a delimiter other than a comma is used in a DIMENSION statement, or (3) there are unmatched parentheses in an array declaration.

Originating Subroutine: PRSPEC

Message:

ERROR POPPING STACK ON ENDDO

Explanation: An ENDDO statement is encountered that matches a corresponding DO or DOWHILE statement with a label reference.

Originating Subroutine: PRSTRC

Message:

*****PRSUBS WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: An ENTRY keyword is not followed by an entry point name.

Originating Subroutine: PRSUBS

Message:

*****PRTOKE WARNING***** - SYNTAX ERROR IN SOURCE

Explanation: A token that is not a constant starts with a character other than a letter, a dollar sign, or a percent sign.

Originating Subroutine: PRTOKE

Message:

*****READER WARNING***** - INPUT CARD LENGTH GREATER THAN 1440

Explanation: A statement containing more than 1440 characters has been encountered.

Originating Subroutine: READER

Message:

****USRWTS: ERROR READING USER WEIGHTS FILE

Explanation: An error occurred while reading the user's weights file.

Originating Subroutine: USRWTS

Message:

****USRWTS: ERROR OPENING DEFAULT WEIGHTS FILE

Explanation: An error occurred while opening the weights file containing the default weights.

Originating Subroutine: USRWTS

Message:

****USRWTS: ERROR READING DEFAULT WEIGHTS FILE

Explanation: An error occurred while reading the default weights file.

Originating Subroutine: USRWTS

REFERENCES

1. American National Standards Institute, ANSI X3.9-1978, American National Standard Programming Language FORTRAN, April 1978

2. Digital Equipment Corporation, AA-1855D-TC, PDP-11 FORTRAN Language Reference Manual, December 1979

3. --, AA-1884C-TC, FORTRAN IV-PLUS User's Guide, December 1979

4. --, AA-D034B-TE, VAX-11 FORTRAN Language Reference Manual, April 1980

5. --, AA-D035B-TE, VAX-11 FORTRAN User's Guide, April 1980

6. International Business Machines Corporation, SC26-3985-3, VS FORTRAN Applications Programming: Guide, September 1982

7. --, GC26-3986-2, VS FORTRAN Applications Programming: Language Reference, September 1982

8. Software Engineering Laboratory, SEL-77-003, Structured FORTRAN Preprocessor (SFORT), B. Chu, D. Wilson, and R. Beard, September 1977

9. M. Halstead, Elements of Software Science. New York: Elsevier Publishing Co., 1977

10. T. J. McCabe, "A Complexity Measure," IEEE Transactions on Software Engineering, December 1976, vol. 2, no. 4, pp. 308-320

11. International Business Machines Corporation, GC26-3902-1, OS/VS2 MVS Utilities, September 1983

12. --, GC28-0692-4, OS/VS2 JCL, May 1979

13. Software Engineering Laboratory, SEL-82-102, FORTRAN Static Source Code Analyzer Program (SAP) System Description, W. Decker, W. Taylor, et al., March 1985

14. --, SEL-81-002, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide, D. Wyckoff, G. Page, F. McGarry, et al., September 1981

15. J. Stroud, "The Fine Structure of Psychological Time," Annals of New York Academy of Sciences, 1966

16. N. Bulut, M. Halstead, and R. Bayer, "Experimental Validation of A Structural Property of FORTRAN Algorithms," Proceedings of the ACM Annual Conference, November 1974, pp. 207-211

STANDARD BIBLIOGRAPHY OF SEL LITERATURE

The technical papers, memorandums, and documents listed in this bibliography are organized into two groups. The first group is composed of documents issued by the Software Engineering Laboratory (SEL) during its research and development activities. The second group includes materials that were published elsewhere but pertain to SEL activities.

SEL-ORIGINATED DOCUMENTS

SEL-76-001, Proceedings From the First Summer Software Engineering Workshop, August 1976

SEL-77-001, The Software Engineering Laboratory, V. R. Basili, M. V. Zelkowitz, F. E. McGarry, et al., May 1977

SEL-77-002, Proceedings From the Second Summer Software Engineering Workshop, September 1977

SEL-77-003, Structured FORTRAN Preprocessor (SFORT), B. Chu and D. S. Wilson, September 1977

SEL-77-004, GSFC NAVPAK Design Specifications Languages Study, P. A. Scheffer and C. E. Velez, October 1977

SEL-78-001, FORTRAN Static Source Code Analyzer (SAP) Design and Module Descriptions, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, February 1978

SEL-78-003, Evaluation of Draper NAVPAK Software Design, K. Tasaki and F. E. McGarry, June 1978

SEL-78-004, Structured FORTRAN Preprocessor (SFORT) PDP-11/70 User's Guide, D. S. Wilson and B. Chu, September 1978

SEL-78-005, Proceedings From the Third Summer Software Engineering Workshop, September 1978

SEL-78-006, GSFC Software Engineering Research Requirements Analysis Study, P. A. Scheffer and C. E. Velez, November 1978

SEL-78-007, Applicability of the Rayleigh Curve to the SEL Environment, T. E. Mapp, December 1978

SEL-78-202, FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 2), W. J. Decker and W. A. Taylor, April 1985

SEL-79-001, SIMPL-D Data Base Reference Manual, M. V. Zelkowitz, July 1979

SEL-79-002, The Software Engineering Laboratory: Relationship Equations, K. Freburger and V. R. Basili, May 1979

SEL-79-003, Common Software Module Repository (CSMR) System Description and User's Guide, C. E. Goorevich, A. L. Green, and S. R. Waligora, August 1979

SEL-79-004, Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment, C. E. Goorevich, A. L. Green, and W. J. Decker, September 1979

SEL-79-005, Proceedings From the Fourth Summer Software Engineering Workshop, November 1979

SEL-80-001, Functional Requirements/Specifications for Code 580 Configuration Analysis Tool (CAT), F. K. Banks, A. L. Green, and C. E. Goorevich, February 1980

SEL-80-002, Multi-Level Expression Design Language-Requirement Level (MEDL-R) System Evaluation, W. J. Decker and C. E. Goorevich, May 1980

SEL-80-003, Multimission Modular Spacecraft Ground Support Software System (MMS/GSSS) State-of-the-Art Computer Systems/Compatibility Study, T. Welden, M. McClellan, and P. Liebertz, May 1980

SEL-80-005, A Study of the Musa Reliability Model, A. M. Miller, November 1980

SEL-80-006, Proceedings From the Fifth Annual Software Engineering Workshop, November 1980

SEL-80-007, An Appraisal of Selected Cost/Resource Estimation Models for Software Systems, J. F. Cook and F. E. McGarry, December 1980

SEL-80-104, Configuration Analysis Tool (CAT) System Description and User's Guide (Revision 1), W. Decker and W. Taylor, December 1982

SEL-81-008, Cost and Reliability Estimation Models (CAREM) User's Guide, J. F. Cook and E. Edwards, February 1981

SEL-81-009, Software Engineering Laboratory Programmer Workbench Phase 1 Evaluation, W. J. Decker and F. E. McGarry, March 1981

SEL-81-011, Evaluating Software Development by Analysis of Change Data, D. M. Weiss, November 1981

SEL-81-012, The Rayleigh Curve As a Model for Effort Distribution Over the Life of Medium Scale Software Systems, G. O. Picasso, December 1981

SEL-81-013, Proceedings From the Sixth Annual Software Engineering Workshop, December 1981

SEL-81-014, Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL), A. L. Green, W. J. Decker, and F. E. McGarry, September 1981

SEL-81-101, Guide to Data Collection, V. E. Church, D. N. Card, F. E. McGarry, et al., August 1982

SEL-81-102, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide Revision 1, P. Lo and D. Wyckoff, July 1983

SEL-81-104, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., February 1982

SEL-81-106, Software Engineering Laboratory (SEL) Document Library (DOCLIB) System Description and User's Guide (Revision 1), W. Taylor and W. J. Decker, May 1985

SEL-81-107, Software Engineering Laboratory (SEL) Compendium of Tools, W. J. Decker, W. A. Taylor, and E. J. Smith, February 1982

SEL-81-110, Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics, G. Page and F. McGarry, December 1983

SEL-81-203, Software Engineering Laboratory (SEL) Data Base Maintenance System (DBAM) User's Guide and System Description, P. Lo, June 1984

SEL-81-205, Recommended Approach to Software Development, F. E. McGarry, G. Page, S. Eslinger, et al., April 1983

SEL-82-001, Evaluation of Management Measures of Software Development, G. Page, D. N. Card, and F. E. McGarry, September 1982, vols. 1 and 2

SEL-82-003, Software Engineering Laboratory (SEL) Data Base Reporting Software User's Guide and System Description, P. Lo, September 1982

SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982

SEL-82-007, Proceedings From the Seventh Annual Software Engineering Workshop, December 1982

SEL-82-008, Evaluating Software Development by Analysis of Changes: The Data From the Software Engineering Laboratory, V. R. Basili and D. M. Weiss, December 1982

SEL-82-102, FORTRAN Static Source Code Analyzer Program (SAP) System Description (Revision 1), W. A. Taylor and W. J. Decker, April 1985

SEL-82-105, Glossary of Software Engineering Laboratory Terms, T. A. Babst, F. E. McGarry, and M. G. Rohleder, October 1983

SEL-82-206, Annotated Bibliography of Software Engineering Laboratory Literature, D. N. Card, Q. L. Jordan, and F. E. McGarry, November 1984

SEL-83-001, An Approach to Software Cost Estimation, F. E. McGarry, G. Page, D. N. Card, et al., February 1984

SEL-83-002, Measures and Metrics for Software Development, D. N. Card, F. E. McGarry, G. Page, et al., March 1984

SEL-83-003, Collected Software Engineering Papers: Volume II, November 1983

SEL-83-006, Monitoring Software Development Through Dynamic Variables, C. W. Doerflinger, November 1983

SEL-83-007, Proceedings From the Eighth Annual Software Engineering Workshop, November 1983

SEL-83-104, Software Engineering Laboratory (SEL) Data Base Retrieval System (DARES) User's Guide, T. A. Babst, W. J. Decker, P. Lo, and W. Miller, August 1984

SEL-83-105, Software Engineering Laboratory (SEL) Data Base Retrieval System (DARES) System Description, P. Lo, W. J. Decker, and W. Miller, August 1984

SEL-84-001, Manager's Handbook for Software Development, W. W. Agresti, V. E. Church, and F. E. McGarry, April 1984

SEL-84-002, Configuration Management and Control: Policies and Procedures, Q. L. Jordan and E. Edwards, December 1984

SEL-84-003, Investigation of Specification Measures for the Software Engineering Laboratory (SEL), W. Agresti, V. Church, and F. E. McGarry, December 1984

SEL-85-001, A Comparison of Software Verification Techniques, D. Card, R. Selby, F. McGarry, et al., April 1985

SEL-RELATED LITERATURE

Agresti, W. W., Definition of Specification Measures for the Software Engineering Laboratory, Computer Sciences Corporation, CSC/TM-84/6085, June 1984

[1]Agresti, W. W., F. E. McGarry, D. N. Card, et al., "Measuring Software Technology," Program Transformation and Programming Environments. New York: Springer-Verlag, 1984

[2]Bailey, J. W., and V. R. Basili, "A Meta-Model for Software Development Resource Expenditures," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

[2]Basili, V. R., "Models and Metrics for Software Management and Engineering," ASME Advances in Computer Technology, January 1980, vol. 1

Basili, V. R., "SEL Relationships for Programming Measurement and Estimation," University of Maryland, Technical Memorandum, October 1979

Basili, V. R., Tutorial on Models and Metrics for Software Management and Engineering. New York: Computer Societies Press, 1980 (also designated SEL-80-008)

[2]Basili, V. R., and J. Beane, "Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problems?", Journal of Systems and Software, February 1981, vol. 2, no. 1

[2]Basili, V. R., and K. Freburger, "Programming Measurement and Estimation in the Software Engineering Laboratory," Journal of Systems and Software, February 1981, vol. 2, no. 1

[1]Basili, V. R., and B. T. Perricone, "Software Errors and Complexity: An Empirical Investigation," Communications of the ACM, January 1984, vol. 27, no. 1

[2]Basili, V. R., and T. Phillips, "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," Proceedings of the ACM SIGMETRICS Symposium/ Workshop: Quality Metrics, March 1981

[1]Basili, V. R., R. W. Selby, and T. Phillips, "Metric Analysis and Data Validation Across FORTRAN Projects," IEEE Transactions on Software Engineering, November 1983

Basili, V. R., and J. Ramsey, Structural Coverage of Functional Testing, University of Maryland, Technical Report TR-1442, September 1984

Basili, V. R., and R. Reiter, "Evaluating Automatable Measures for Software Development," Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity and Cost, October 1979

[1]Basili, V.R., and D. M. Weiss, A Methodology for Collecting Valid Software Engineering Data, University of Maryland, Technical Report TR-1235, December 1982

Basili, V. R., and M. V. Zelkowitz, "Designing a Software Measurement Experiment," Proceedings of the Software Life Cycle Management Workshop, September 1977

[2]Basili, V. R., and M. V. Zelkowitz, "Operation of the Software Engineering Laboratory," Proceedings of the Second Software Life Cycle Management Workshop, August 1978

[2]Basili, V. R., and M. V. Zelkowitz, "Measuring Software Development Characteristics in the Local Environment," Computers and Structures, August 1978, vol. 10

Basili, V. R., and M. V. Zelkowitz, "Analyzing Medium Scale Software Development," Proceedings of the Third International Conference on Software Engineering. New York: Computer Societies Press, 1978

[2]Basili, V. R., and M. V. Zelkowitz, "The Software Engineering Laboratory: Objectives," Proceedings of the Fifteenth Annual Conference on Computer Personnel Research, August 1977

[2]Chen, E., and M. V. Zelkowitz, "Use of Cluster Analysis To Evaluate Software Engineering Methodologies," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

[1]Doerflinger, C. W., and V. R. Basili, "Monitoring Software Development Through Dynamic Variables," Proceedings of the Seventh International Computer Software and Applications Conference. New York: Computer Societies Press, 1983

Higher Order Software, Inc., TR-9, A Demonstration of AXES for NAVPAK, M. Hamilton and S. Zeldin, September 1977 (also designated SEL-77-005)

Page, G., F. E. McGarry, and D. N. Card, "A Practical Experience With Independent Verification and Validation," Proceedings of the Eighth International Computer Software and Applications Conference, November 1984

Turner, C., and G. Caron, A Comparison of RADC and NASA/SEL Software Development Data, Data and Analysis Center for Software, Special Publication, May 1981

Turner, C., G. Caron, and G. Brement, NASA/SEL Data Compendium, Data and Analysis Center for Software, Special Publication, April 1981

[2]Zelkowitz, M. V., "Resource Estimation for Medium Scale Software Projects," Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science. New York: Computer Societies Press, 1979

[1]Zelkowitz, M. V., "Data Collection and Evaluation for Experimental Computer Science Research," Empirical Foundations for Computer and Information Science (proceedings), November 1982

Zelkowitz, M. V., and V. R. Basili, "Operational Aspects of a Software Measurement Facility," Proceedings of the Software Life Cycle Management Workshop, September 1977

[1]This article also appears in SEL-83-003, Collected Software Engineering Papers: Volume II, November 1983.

[2]This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume I, July 1982.